# Llave electrónica basada en patrones de movimiento

**Ingeniería Técnica de Informática de Sistemas**

**Estudiante**
**Ramón González Valle**

**Consultor**
**Jordi Bécares Ferrés**

Fecha de entrega
21/05/2014

# *Dedicatoria y agradecimientos*

Si bien agradezco a todos los profesores, consultores y compañeros que he tenido a lo largo del tiempo el haber llegado a lo que es la culminación de una fase, quiero hacer una mención especial a Alicia, profesora de electrónica en mis tiempos de estudiante en el instituto Comte de Rius de Tarragona, que me alentó a continuar.

Pero sobre todo agradezco y quiero dedicar este proyecto, por lo que significa y no por lo que es, a mi hijo y a mi mujer, por el tiempo que les he robado y por la sobrecarga de tareas que asumió mi mujer para que yo pudiera llegar a este punto.

# *Resumen*

Este proyecto corresponde al trabajo de final de carrera de Ingeniería Técnica de Informática de Sistemas, dentro del área de Sistemas Embebidos. En esta área se pretende desarrollar una aplicación práctica para dichos sistemas embebidos.

El proyecto desarrollado consiste en un sistema capaz de registrar, comparar y reconocer patrones de movimiento

El sistema funciona como un gestor de claves electrónicas. Permite almacenar varias claves generadas mediante el mismo sistema y compararlas posteriormente con una introducida por un usuario. Si la clave introducida corresponde con una de las almacenadas enviará un mensaje a un host conectado en una red Ethernet, que puede ser local o de Internet, indicando dicha coincidencia. Cada una de estas claves es un patrón de movimiento digitalizado por medio del acelerómetro

Mediante este sistema se pueden llegar a implementar desde un control de permisos de acceso hasta un control remoto. Para el primer caso los movimientos corresponderían a claves de acceso, donde el sistema reconocería al usuario por el movimiento realizado y en el segundo caso cada uno de movimientos correspondería a las diferentes acciones que se desea realizar en el sistema controlado. En ambos casos los patrones a reconocer podrían estar pregrabados o permitir que el propio usuario fuera el que los definiera.

El hecho de que la interacción con el usuario se pueda simplificar a que éste realice una serie de movimientos con el gadget, cada uno de los cuales puede ser tan simple como se desee, permite que pueda ser utilizado por personas de muy diversas edades y limitaciones, tanto físicas como mentales.

# Índice de contenidos


**1 Introducción ..... 7**
- 1.1 Justificación ..... 7
- 1.2 Descripción ..... 7
- 1.3 Objetivos ..... 8
- 1.4 Enfoque y metodología seguida ..... 9
- 1.5 Planificación ..... 10
  - 1.5.1 Planificación inicial ..... 10
  - 1.5.2 Resultado final ..... 11
- 1.6 Recursos empleados ..... 12
  - 1.6.1 Hardware ..... 12
  - 1.6.2 Software ..... 14
- 1.7 Productos obtenidos ..... 14

**2 Antecedentes ..... 15**
- 2.1 Estudio de mercado ..... 15

**3 Descripción funcional ..... 16**
- 3.1 Llave electrónica basada en patrones de movimiento ..... 16
- 3.2 Host ..... 16
- 3.3 PC usuario ..... 16
- 3.4 Sistema embebido ..... 17

**4 Descripción detallada ..... 20**
- 4.1 Esquema de conexionado ..... 20
- 4.2 Módulos sistema embebido ..... 21
  - 4.2.1 Inicio del sistema ..... 21
  - 4.2.2 Tarea del interface con el usuario ..... 23
  - 4.2.3 Tarea de gestión de la WLAN ..... 24
  - 4.2.4 Tarea cliente TCP ..... 26
  - 4.2.5 Tarea de gestión de patrones ..... 28
  - 4.2.6 Tarea de gestión de LEDs y pulsadores ..... 30
  - 4.2.7 Gestión de memoria no volátil ..... 30
- 4.3 Patrones de movimiento ..... 33
  - 4.3.1 Conceptos previos ..... 33
  - 4.3.2 Identificación de un movimiento ..... 35
  - 4.3.3 Estructura de un patrón ..... 40
  - 4.3.4 Adaptación de las señales del acelerómetro ..... 41
  - 4.3.5 Comparación de patrones ..... 43

**5 Conclusiones ..... 44**
- 5.1 Propuesta de mejoras ..... 44
- 5.2 Autoevaluación ..... 44

**6 Glosario ..... 46**

**7 Referencias y bibliografía ..... 47**

**8 Anexos ..... 48**
- 8.1 Manual de usuario ..... 48
  - 8.1.1 Requisitos iniciales ..... 48
  - 8.1.2 Configuración ..... 48
  - 8.1.3 Edición de patrones ..... 50
  - 8.1.4 Entrada de una clave de movimiento ..... 52
  - 8.1.5 LEDs indicativos ..... 52

*8.2 Ejecución y compilación ..... 52*
*8.2.1 Importación del proyecto ..... 53*
*8.2.2 Compilación del proyecto ..... 56*
*8.2.3 Cargar del proyecto ..... 57*

# *Índice de figuras*

*Figura 1. Planificación inicial* ..... 10
*Figura 2. Resultado final de la planificación* ..... 11
*Figura 3. Hardware utilizado* ..... 12
*Figura 4. Diagrama de bloques general del sistema* ..... 16
*Figura 5. Diagrama de bloques del sistema embebido* ..... 17
*Figura 6. Esquema de conexionado* ..... 20
*Figura 7. Diagrama de flujo Inicio del sistema (1)* ..... 21
*Figura 8. Diagrama de flujo Inicio del sistema (2)* ..... 22
*Figura 9. Diagrama de flujo tarea interface con el usuario* ..... 23
*Figura 10. Diagrama de flujo tarea de gestión de la WLAN (1)* ..... 24
*Figura 11. Diagrama de flujo tarea de gestión de la WLAN (2)* ..... 25
*Figura 12. Diagrama de flujo tarea de gestión del cliente TCP (1)* ..... 26
*Figura 13. Diagrama de flujo tarea de gestión del cliente TCP (2)* ..... 27
*Figura 14. Diagrama de flujo tarea de gestión de patrones (1)* ..... 28
*Figura 15. Diagrama de flujo tarea de gestión de patrones (2)* ..... 29
*Figura 16. Mecanismo básico de la gestión de la memoria flash interna* ..... 32
*Figura 17. Estado de los sectores en la gestión de la memoria flash interna* ..... 32
*Figura 18. Leyes de desplazamiento, velocidad y aceleración de un elemento con movimiento de traslación con arranque y parada* ..... 34
*Figura 19. Aceleración en un movimiento de ida y vuelta* ..... 35
*Figura 20. Detección de movimientos* ..... 36
*Figura 21. Detección de movimientos en dos ejes en forma independiente* ..... 37
*Figura 22. Detección de nuevo movimiento en un eje* ..... 38
*Figura 23. Detección global de nuevo movimiento* ..... 38
*Figura 24. Detección de movimientos en dos ejes en forma conjunta, consideración de un solo movimiento* ..... 39
*Figura 25. Detección de movimientos en dos ejes en forma conjunta, consideración de dos movimientos* ..... 40
*Figura 26. Estructura de un patrón de movimiento* ..... 41
*Figura 27. Filtro tratamiento señal acelerómetro* ..... 42
*Figura 28. Menú general* ..... 48
*Figura 29. Menú configuración WIFI* ..... 49
*Figura 30. Menú configuración Cliente TCP* ..... 49
*Figura 31. Menú patrones de movimiento* ..... 50
*Figura 32. Edición de los patrones almacenados* ..... 50
*Figura 33. Creación de un patrón de movimiento* ..... 51
*Figura 34. Asignación de nombre para un patrón* ..... 51
*Figura 35. Importar proyecto LPCXpresso (1)* ..... 53
*Figura 36. Importar proyecto LPCXpresso (2)* ..... 53
*Figura 37. Importar proyecto LPCXpresso (3)* ..... 54
*Figura 38. Importar proyecto LPCXpresso (4)* ..... 54
*Figura 39. Importar proyecto LPCXpresso (5)* ..... 55
*Figura 40. Importar proyecto LPCXpresso (6)* ..... 55
*Figura 41. Compilar proyecto LPCXpresso (1)* ..... 56
*Figura 42. Compilar proyecto LPCXpresso (2)* ..... 56
*Figura 43. Cargar el proyecto en el LPC (1)* ..... 57
*Figura 44. Cargar el proyecto en el LPC (2)* ..... 57
*Figura 45. Cargar el proyecto en el LPC (3)* ..... 58
*Figura 46. Indicación de final de cargar del proyecto en el LPC* ..... 58

# 1 Introducción

## 1.1 Justificación

El control de acceso es un tema que evoluciona constantemente, reconocimiento de huella dactilar, de retina, facial,…, sistemas pensados para el reconocimiento individual. El sistema presentado permite crear un control de acceso que se puede considerar tan válido como el de la introducción de una clave escrita, ya que permite que se creen claves personalizadas e identificarlas individualmente con un coste, complejidad y requerimiento menor a otros.

Por otro lado, cada vez más los sistemas inteligentes, más conocidos por su palabra inglesa *smart*, se están integrando en nuestro quehacer de cada día. Si bien en algunos casos dicha denominación no es más que un gancho publicitario, es cierto que cada vez son más los dispositivos que poseen funcionalidades ampliadas (neveras que llevan el stock de productos, relojes con captación de biorritmos, televisores que integran la conectividad a internet y mezcla su contenido con el de la propia televisión,…). El efecto directo de la ampliación de las funcionalidades es que también se amplían las opciones que ha de configurar o a las que ha de acceder el usuario, con lo que su manejo se suele complicar.

El propio avance de estos sistemas también proporciona métodos de interacción más naturales, como son los mandos giroscópicos, la detección de movimientos con cámaras,… Dentro de éstos es donde se puede englobar una de las funcionalidades del sistema presentado, ya que pude permitir el envío de comandos generados por simples movimientos.

## 1.2 Descripción

Este sistema permite almacenar patrones de movimientos y su posterior reconocimiento. Al estar conectado a una WLAN, a través del módulo WIFI que incorpora, cada vez que reconoce uno de los patrones almacenados envía un mensaje a un host predefinido, que puede local o remoto.

Dado que este sistema sólo se encarga del procesamiento de los patrones, a nivel de almacenamiento y reconocimiento y no de la actuación a realizar cuando encuentra una coincidencia, simplemente envía un mensaje indicando dicha situación, puede ser utilizado tanto para permisos de acceso como para el control de otros sistemas mediante comandos de movimiento.

Procesa los tres ejes, por lo que los patrones de movimiento son tridimensionales.

A través de una interface basada en un terminal en el PC el usuario puede indicar que quiere crear un patrón y asignarle un nombre identificativo, tomará el gadget y accionará un pulsador en el propio gadget para indicar el inicio de la introducción del patrón. La liberación de dicho pulsador indicará que ha finalizado la introducción, instante en que quedará almacenado en el sistema. Mediante la misma interface se pueden eliminar patrones, referenciadas por su nombre identificativo asociado.

Si el usuario simplemente coge el gadget y acciona el pulsador el sistema considerará que está introduciendo un movimiento, al liberar el pulsador lo comparará con los patrones almacenados, si encuentra alguna coincidencia enviará un mensaje al host configurado, indicando el nombre identificativo asociado al patrón reconocido.

Mediante la interface también se configura la red WIFI a la que se conecta el sistema, los parámetros de la LAN (IP del equipo, puerta de enlace y servidor DNS) y el host al que se le enviarán los mensajes.

Dispone de unos LEDs que indican el estado general del sistema, el estado de la conexión WLAN, que se está en proceso de captura de un patrón y que se está en proceso de envío de un mensaje al host.

## 1.3 Objetivos

Los objetivos de este sistema son los siguientes:

- **Interactuar con el usuario mediante un interface tipo terminal**
  Se emitirán mensajes al terminal conectado y permitirá, mediante un simple menú, la navegación e introducción de datos para configurar el sistema.
- **Proporcionar comunicación WIFI con otros sistemas**
  Se conectará a la WLAN deseada, permitiendo configurar tanto los parámetros necesarios para conectarse a la WIFI (SSID, clave) como los de la LAN (IP, servidor DNS, puerta de enlace).
- **Adquirir datos digitales y analógicos en tiempo real**
  Leerá el estado de una entrada digital conectada a un pulsador y de tres entradas analógicas conectadas a cada uno de los ejes del acelerómetro, todas ellas simultáneamente.
- **Activar señales digitales en tiempo real**
  Activará varias salidas digitales conectadas a LEDs que indicarán ciertos estados del sistema.
- **Acceder a memoria no volátil para almacenar y recuperar datos**
  Gestionará el uso de memoria no volátil del propio microcontrolador donde almacenará los datos que interesa que el sistema no pierda ante la supresión de la tensión de alimentación.
- **Dotar de unos algoritmos que permitan el almacenaje e interpretación, por medio de la comparación, de patrones de movimiento**
  Con los valores de los ejes del acelerómetro, generará una *imagen digital* del patrón de movimiento introducido, permitiendo su almacenamiento y comparación.
- **Enviar un mensaje a un host en caso de que se introduzca un movimiento que corresponda con alguno de los patrones almacenados**
  Mediante la conexión a la WLAN establecida se conectará y enviará un mensaje al host configurado.
- **Recuperación de las comunicaciones ante pérdidas de cobertura o de conexión**
  Ante un fallo de conexión con la WLAN intentará reconectar automáticamente. Cuando tenga que enviar un mensaje al host, si la WLAN está activa, intentará conectar con él varias veces antes de considerar que no está accesible.

## 1.4 Enfoque y metodología seguida

Para la realización del presente proyecto he seguido las siguientes fases:

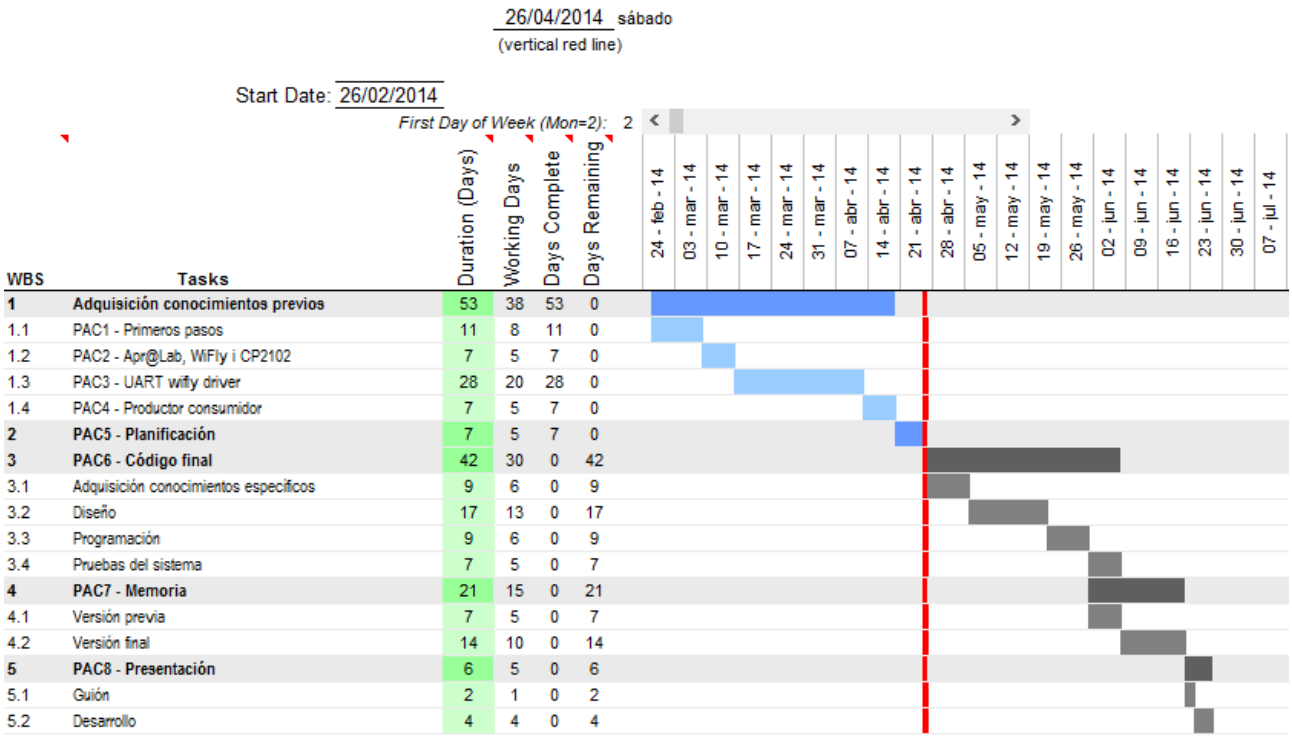
- **Adquisición de conocimientos previos sobre los elementos generales del gadget**
  Inicialmente, mediantes la realización de las PACs, he adquirido conocimientos previos básicos:
  - Entorno de programación.
    Conocer el entorno LPCXpresso y el sistema operativo FreeRTOS, utilizados para la programación del microcontrolador, mediante la programación y compilación de unos proyectos iniciales.
  - LPC 1769.
    Conocer la placa con el microcontrolador ARM Cortex-M3 y el módulo LPC-Link para su programación y debug. He transferido los proyectos iniciales y realizado el debug de alguno de ellos.
  - UART.
    Conocer y utilizar una librería para la gestión de los puertos UART del LPC.
  - Convertidor USB-UART CP2102
    Conocer el módulo, interconectarlo con el LPC y crear unas librerías para utilizarlo como debug e interacción con el sistema, conectándolo a un terminal en el PC.
  - Módulo radio WiFly.
    Conocer el módulo, interconectarlo con el LPC y crear unas librerías para utilizarlo y dotar al sistema de acceso a una WLAN y, a través de él, a servidores externos.
- **Adquisición de conocimientos sobre los elementos específicos necesarios**
  Con la decisión del proyecto concreto a realizar, he tenido que adquirir conocimientos sobre otros elementos que necesitaba utilizar.
  - ADC.
    Conocer y crear una librería para gestionar el conversor analógico-digital del LPC.
  - Entradas/Salidas digitales.
    Conocer y crear una librería para gestionar entradas/salidas del LPC.
  - Acelerómetro MMA7361.
    Conocer el módulo, interconectarlo con el LPC y crear unas librerías para utilizarlo y acceder a su información.
  - Acceso lectura/escritura a memoria no volátil.
    Evaluar qué memoria no volátil utilizar (interna o externa al microcontrolador) y utilizar una librería para gestionarla.
- **Representación digital de un patrón de movimiento**
  - Análisis y estudio de la problemática.
    Analizar y estudiar qué parámetros puedo tomar como representativos de un movimiento, a través del acelerómetro, con un bajo coste computacional.
  - Diseño de la representación y comparación.
    Diseñar cómo representar digitalmente un patrón de movimiento de una forma compacta e inequívoca, apta para un bajo coste de almacenamiento y una comparación sencilla.

- **Programación**.
  - Programación del diseño de la representación del patrón de movimiento.
    Programar el diseño realizado en el lenguaje de programación del LPC.
  - Programación de la totalidad del proyecto.
    Programar el proyecto, uniendo los diversos módulos, para generar el programa definitivo, compilándolo y enviándolo al LPC.
- **Pruebas del sistema**.
  Probar el sistema completo, con todas las funcionalidades previstas, corrigiendo los errores detectados.
- **Realización de la memoria**.
  Realizar la presente memoria del proyecto.

## 1.5 Planificación

### 1.5.1 Planificación inicial

Inicialmente he partido con la siguiente planificación:

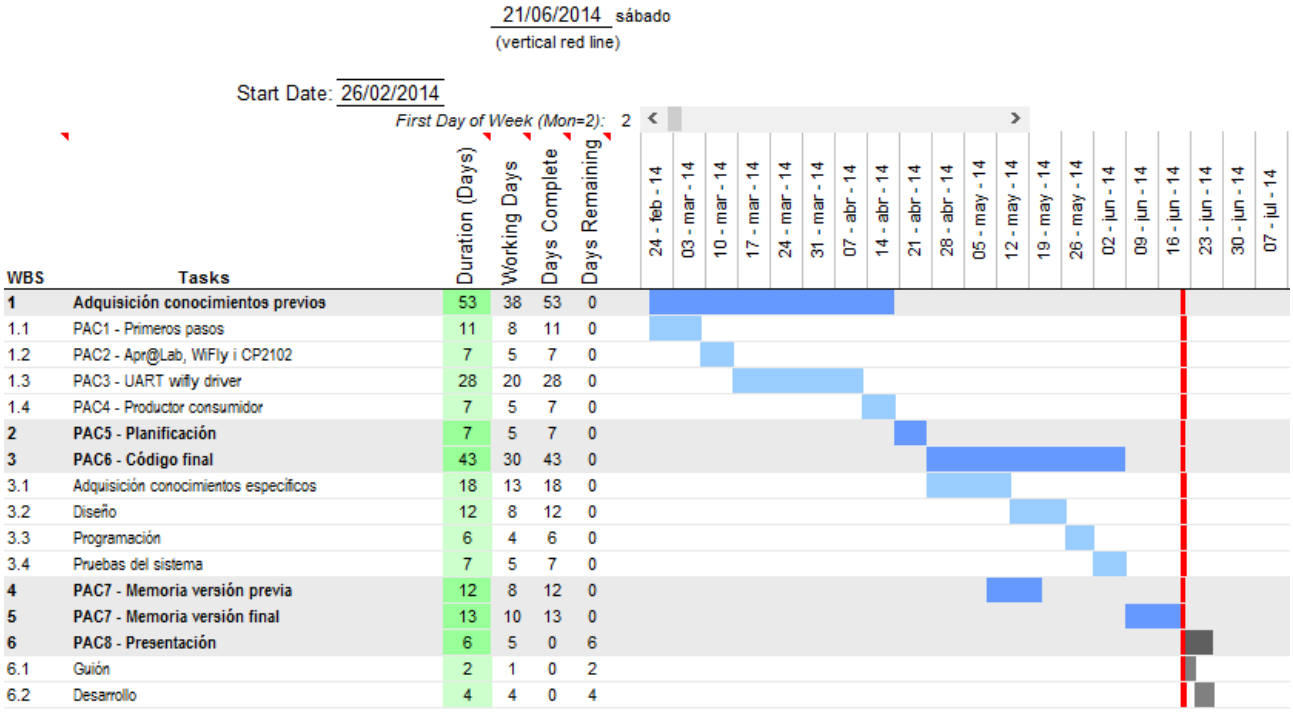



| WBS | Tasks | Duration (Days) | Working Days | Days Complete | Days Remaining |
|---|---|---|---|---|---|
| **1** | **Adquisición conocimientos previos** | 53 | 38 | 53 | 0 |
| 1.1 | PAC1 - Primeros pasos | 11 | 8 | 11 | 0 |
| 1.2 | PAC2 - Apr@Lab, WiFly i CP2102 | 7 | 5 | 7 | 0 |
| 1.3 | PAC3 - UART wifly driver | 28 | 20 | 28 | 0 |
| 1.4 | PAC4 - Productor consumidor | 7 | 5 | 7 | 0 |
| **2** | **PAC5 - Planificación** | 7 | 5 | 7 | 0 |
| **3** | **PAC6 - Código final** | 42 | 30 | 0 | 42 |
| 3.1 | Adquisición conocimientos específicos | 9 | 6 | 0 | 9 |
| 3.2 | Diseño | 17 | 13 | 0 | 17 |
| 3.3 | Programación | 9 | 6 | 0 | 9 |
| 3.4 | Pruebas del sistema | 7 | 5 | 0 | 7 |
| **4** | **PAC7 - Memoria** | 21 | 15 | 0 | 21 |
| 4.1 | Versión previa | 7 | 5 | 0 | 7 |
| 4.2 | Versión final | 14 | 10 | 0 | 14 |
| **5** | **PAC8 - Presentación** | 6 | 5 | 0 | 6 |
| 5.1 | Guión | 2 | 1 | 0 | 2 |
| 5.2 | Desarrollo | 4 | 4 | 0 | 4 |

*Figura 1. Planificación inicial*

### 1.5.2 Resultado final

Finalmente el proyecto ha seguido la siguiente ejecución:




| WBS | Tasks | Duration (Days) | Working Days | Days Complete | Days Remaining |
|---|---|---|---|---|---|
| **1** | **Adquisición conocimientos previos** | 53 | 38 | 53 | 0 |
| 1.1 | PAC1 - Primeros pasos | 11 | 8 | 11 | 0 |
| 1.2 | PAC2 - Apr@Lab, WiFly i CP2102 | 7 | 5 | 7 | 0 |
| 1.3 | PAC3 - UART wifly driver | 28 | 20 | 28 | 0 |
| 1.4 | PAC4 - Productor consumidor | 7 | 5 | 7 | 0 |
| **2** | **PAC5 - Planificación** | 7 | 5 | 7 | 0 |
| **3** | **PAC6 - Código final** | 43 | 30 | 43 | 0 |
| 3.1 | Adquisición conocimientos específicos | 18 | 13 | 18 | 0 |
| 3.2 | Diseño | 12 | 8 | 12 | 0 |
| 3.3 | Programación | 6 | 4 | 6 | 0 |
| 3.4 | Pruebas del sistema | 7 | 5 | 7 | 0 |
| **4** | **PAC7 - Memoria versión previa** | 12 | 8 | 12 | 0 |
| **5** | **PAC7 - Memoria versión final** | 13 | 10 | 13 | 0 |
| **6** | **PAC8 - Presentación** | 6 | 5 | 0 | 6 |
| 6.1 | Guión | 2 | 1 | 0 | 2 |
| 6.2 | Desarrollo | 4 | 4 | 0 | 4 |

*Figura 2. Resultado final de la planificación*

Los motivos del desfase han sido:

- Por un error en la lectura del plan docente había planificado la entrega de la versión previa de la memoria en otra fecha.
- Las pruebas con la lectura de los ejes del acelerómetro me hizo darme cuenta que tenía que realizar un tratamiento previo, teniendo que incluir filtros previos.
- Para el almacenamiento en memoria no volátil he utilizado una librería creada por el propio fabricante del microcontrolador, pero lamentablemente tenía errores que impedía su correcta compilación con el entorno utilizado y que no funcionaba correctamente con determinada parametrización, errores que tuve que corregir.

## 1.6 Recursos empleados

### 1.6.1 Hardware

Para el presente proyecto he utilizado los siguientes elementos de hardware:

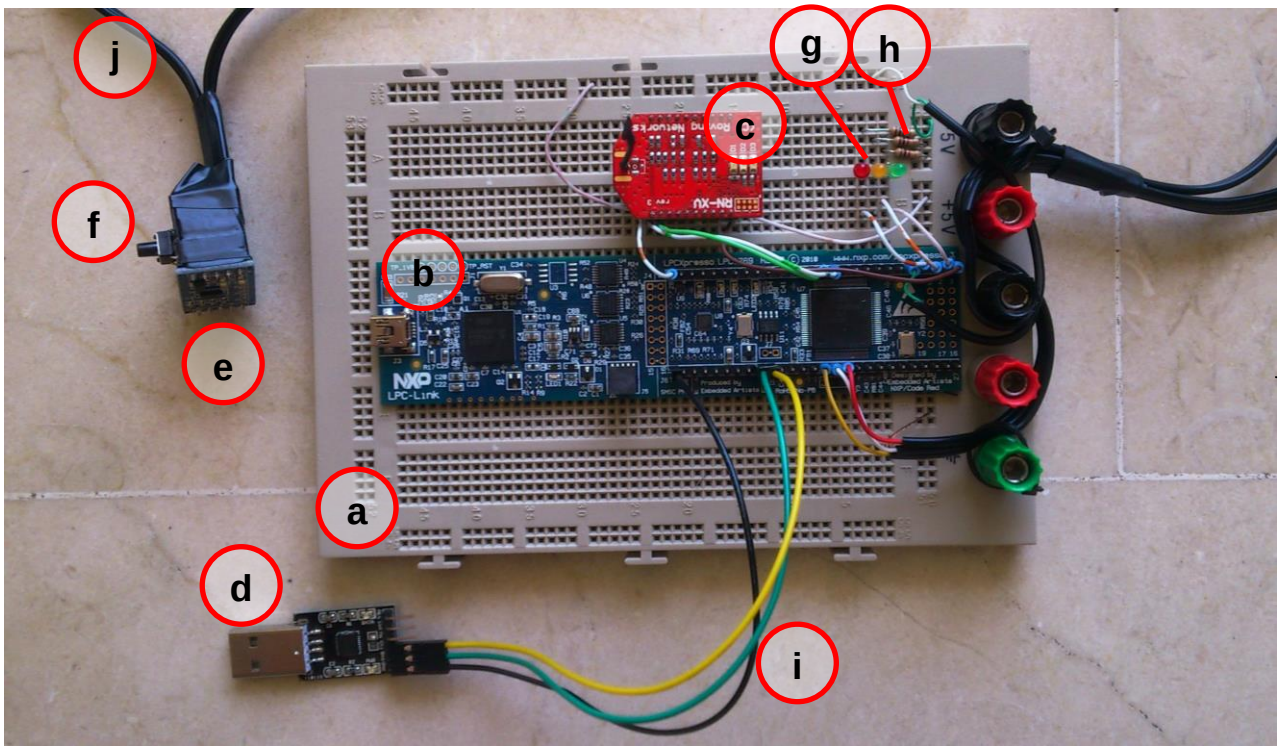



*Figura 3. Hardware utilizado*

**a) Protoboard**

Placa de pruebas para realizar prototipos y poder interconectar componentes electrónicos sin soldadura.

Lo utilizo para realiza el prototipo de una forma más rápida, cómoda y flexible.

**b) LPC 1769**

Sistema embebido de NXP Semiconductors de evaluación basado en un microcontrolador ARM Cortex-M3, que incluye un JTAG debugger (LPC-Link) que permite la conexión a través de USB tanto para la carga del programa en el microcontolador como para su depuración on-chip.

Es el núcleo del sistema, el que contiene al microcontrolador.

**c) WiFly RN-XV**

Módulo radio de Roving Networks, basado en un módulo WIFI RN171, que incorpora un firmware pregrabado el cual permite, a través de su UART, una fácil integración de cualquier sistema en una WLAN.

Es utilizado para conectar a la WLAN y poder enviar los mensajes al host.

**d) USB-UART CP2102**

Bridge que permite la conexión USB a UART. En un PC, mediante la carga de un driver, permite acceder a él al instalar un puerto serie virtual.

Comunica el sistema con el terminal, mediante el que interactúa el usuario con el sistema.

**e) MMA7361**

Módulo sensor analógico acelerómetro e inclinómetro de tres ejes. Dispone de una salida analógica en tensión por cada eje cuyo valor es proporcional a la fuerza G aplicada sobre él.

Es el sensor utilizado para detectar los movimientos.

**f) Pulsador**

Utilizado para que el sistema empieza a capturar los movimientos.

**g) LEDs**

Diodos Emisores de Luz.

Utilizados para indicar diversos estados del sistema, cada uno de un color diferente.

**h) Resistencias**

Utilizadas para limitar la corriente de los LEDs.

**i) Cables interconexión**

Permite conectar los diferentes componentes.

**j) Cable coaxial**

Utilizados para conectar las salidas analógicas del acelerómetro con las entradas analógicas del LPC y disminuir los posibles ruidos electromagnéticos que pueden alterar el valor real de las señales.

### 1.6.2 Software

Para crear el entorno de desarrollo he creado una máquina virtual mediante VMware. De esta forma tengo un sistema limpio, dedicado, ampliable y exportable sin ningún condicionante de ningún software incompatible.

A nivel general, en esta máquina instalé los siguientes elementos de software para el desarrollo y pruebas del sistema:

- SO: Windows 7 Profesional de 32 bits.
- Entorno de programación del LPC: LPCXpresso v7.0.0.
- Lenguaje de programación: C.
- SO integrado en el LPC: FreeRTOS v7.1.0.
- Herramientas auxiliares:
    - Drivers varios necesarios, como los que instala LPCXpresso para el acceso al LPC y el driver del convertidor USB-UART CP2102.
    - Programa terminal PuTTY.
    - Programa terminal, servidor TCP,… Hercules SETUP utility.

A nivel de fuentes de información conté con:

- Wiki de la asignatura.
- Manuales y Web del fabricante del LPC http://www.nxp.com, http://www.lpcware.com,...
- Manuales y Web del desarrollador del SO utilizado en el LPC http://www.freertos.org
- Manuales del WiFly, convertidor USB-UART CP2102, acelerómetro MMA7361.
- Internet en general.

## 1.7 Productos obtenidos

El producto obtenido es este proyecto es el sistema de llave electrónica basada en movimientos, formado por otros subsistemas embebidos los cuales acaban generando un sistema atómico.

Este es un sistema independiente de otros sistemas para su funcionamiento. Únicamente hace uso de un host externo que sería el que podría tomar la decisión de cómo ante un aviso de patrón reconocido, que podría ser de permiso de acceso o de actuación por un comando recibido.

# 2 Antecedentes

## 2.1 Estudio de mercado

Buscando por Internet aplicaciones o sistemas que utilicen los movimientos para realizar algún tipo de actuación o como claves de acceso, no he encontrado ninguna que utilice los movimientos como claves ni en sistemas de seguridad o permisos de acceso.

En el campo del control remoto, hay sistemas que disponen de funcionalidades accesibles a través del movimiento, dentro de estos sistemas puedo destacar:

- **Magus 3D gestos**
  Magus es una app para sistemas operativos Android que le permite mover el teléfono en tres dimensiones para abrir aplicaciones, desbloquear, tomar fotos… Es un control gesticular para sistemas Android que cuenten con un acelerómetro. Como teléfonos móviles y tabletas.

- **Mando de Wii Plus**

  El mando de Wii Plus es un control por movimientos de la consola Wii. Responde al movimiento y la rotación que realices con el mando.

- **PlayStation Move**

  PlayStation Move es un sistema de control de videojuegos por movimientos para la consola PlayStation 3.

# 3 Descripción funcional

## 3.1 Llave electrónica basada en patrones de movimiento

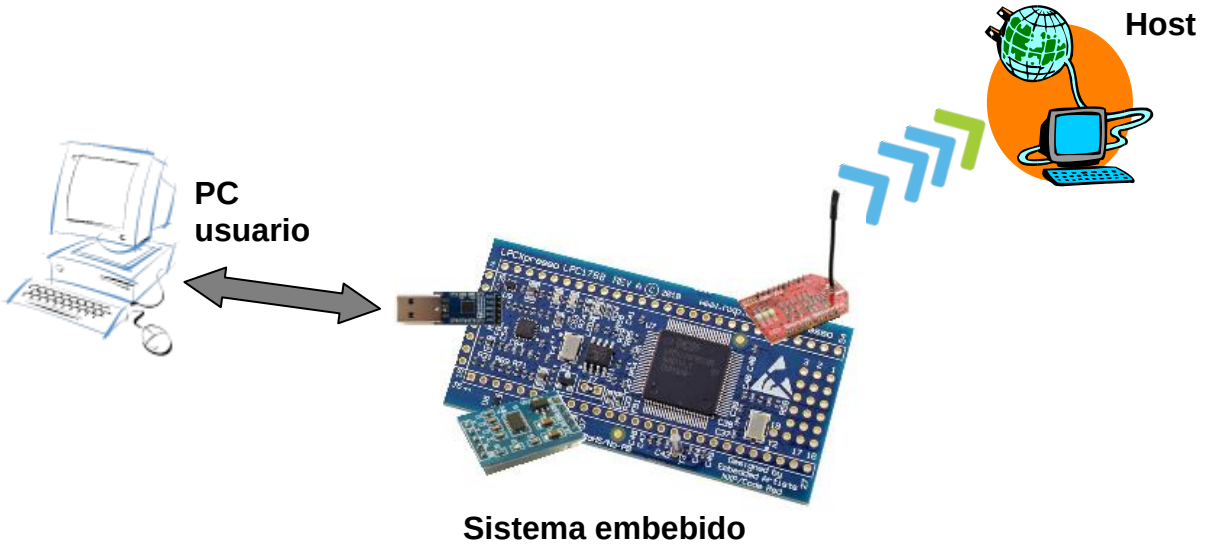



*Figura 4. Diagrama de bloques general del sistema*

El sistema completo está compuesto por un host, que recibirá un mensaje cada vez que el sistema reconozca un patrón de movimiento, un PC de usuario, que permite interactuar con el sistema embebido permitiendo configurarlo y editar patrones y el propio sistema embebido.

La comunicación con el host se realiza vía WIFI. El sistema embebido se conectará a la WLAN configurada, a través de la cual se conectará al host, cuya localización sólo vendrá condicionada por el alcance de la propia red.

La conexión con el PC de usuario ser realiza vía serie a través del convertidor USB-UART CP2102.

## 3.2 Host

El host al que se conecta el sistema, cuando reconoce un patrón, simplemente es un sistema con un servicio TCP escuchando en un puerto determinado. La IP y el puerto de escucha del host son parámetros configurables del sistema.

## 3.3 PC usuario

El PC de usuario únicamente ha de tener instalado el driver del convertidor USB-UART CP2102, el cual instalará un puerto serie virtual, y un programa terminal serie para conectarse con el sistema embebido por dicho puerto serie. Los parámetros del puerto serie son: 57600, 8, N, 1.

## 3.4 Sistema embebido

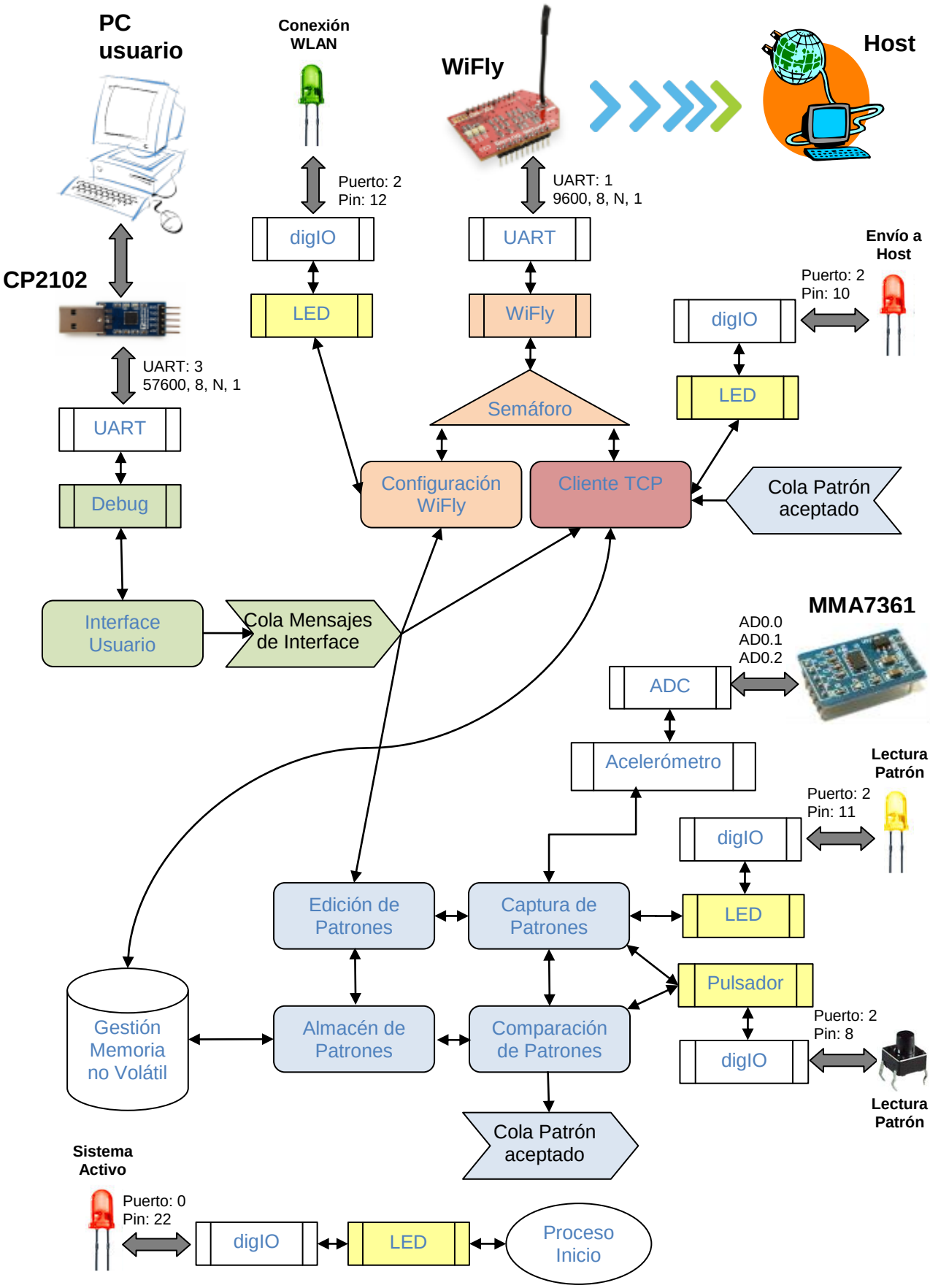



*Figura 5. Diagrama de bloques del sistema embebido*

El sistema se basa en las siguientes tareas principales:

- **Tarea del interface con el usuario**

  Genera un menú en el PC del usuario con el que se puede navegar por la configuración del sistema. Genera el menú principal desde el que se accede a un submenú de configuración del WLAN, un submenú de configuración del cliente TCP y un submenú de edición de los patrones. La comunicación entre esta tarea y el resto se realiza a través de mensajes que envía a una cola, estando codificado en el mensaje la tarea destinataria.

  Cada uno de los submenús ya es gestionado por su propia tarea.

- **Tarea de gestión de la WLAN**

  Configura la WLAN, la WIFI a la que conecta (SSID, clave) y los de la red Ethernet (IP, servidor DNS, puerta de enlace), y vigila que la conexión esté activa, indicándolo a través del LED *Conexión WLAN*. Este LED parpadea cuando está esperando que la conexión esté activa y fijo cuando está activa.

  Inicialmente no intenta configurar la WLAN, por lo que considera que permanecerá con los últimos parámetros asignados. Cuando se le manda configurarlos los graba de forma persistente en el módulo WiFly.

- **Tarea cliente TCP**

  Configurar la IP y puerto de escucha del servidor TCP al que enviarle los mensajes y lee la cola de mensajes que rellena la tarea de gestión de patrones.

  Todo mensaje que recibe es reenviado al servidor configurado, indicándolo que está en proceso de envío través del LED *Envío a Host*, el cual parpadea indicando este estado. Reintenta hasta tres veces el envío del mensaje al servidor TCP, con un tiempo de espera entre reintentos de cinco segundos.

  Por defecto la IP del servidor es 192.168.254.1 y el puerto 5123. Los valores asignados los grabará en la memoria no volátil, de forma que si al iniciarse la tarea encuentra valores guardados los utilizará como válidos en lugar de los de defecto.

- **Tarea de gestión de patrones**

  Permite editar los patrones, listarlos, borrarlos y crearlos. También se encarga de vigilar la entrada de un movimiento, indicado a través del pulsador, para que sea comparado con los almacenados.

  Está limitado a un máximo de diez patrones por el mero hecho de concretar un máximo, ya que el único límite real es la capacidad de memoria no volátil y en este sistema podría haber almacenado más de quinientos simplemente con un simple cambio de parámetros en el código.

Cuando se está introduciendo un movimiento, ya sea para compararlo o para crear un patrón, lo indica a través del LED *Lectura Patrón*, que parpadea indicando este estado.

Cuando se introduce un movimiento que coincide con uno de los patrones almacenados, envía un mensaje a la tarea Cliente TCP a través de una cola. Este mensaje está compuesto por el nombre asignado al patrón reconocido.

- **Tarea de gestión de LEDs y pulsadores**
  Básicamente se encarga de configurar de una forma coherente las entradas asignadas a pulsadores y salidas asignadas a LEDs, de realizar una lectura periódica de los pulsadores y de gestionar el parpadeo de los LEDs para simplicidad del resto de módulos.

# 4 Descripción detallada

## 4.1 Esquema de conexionado

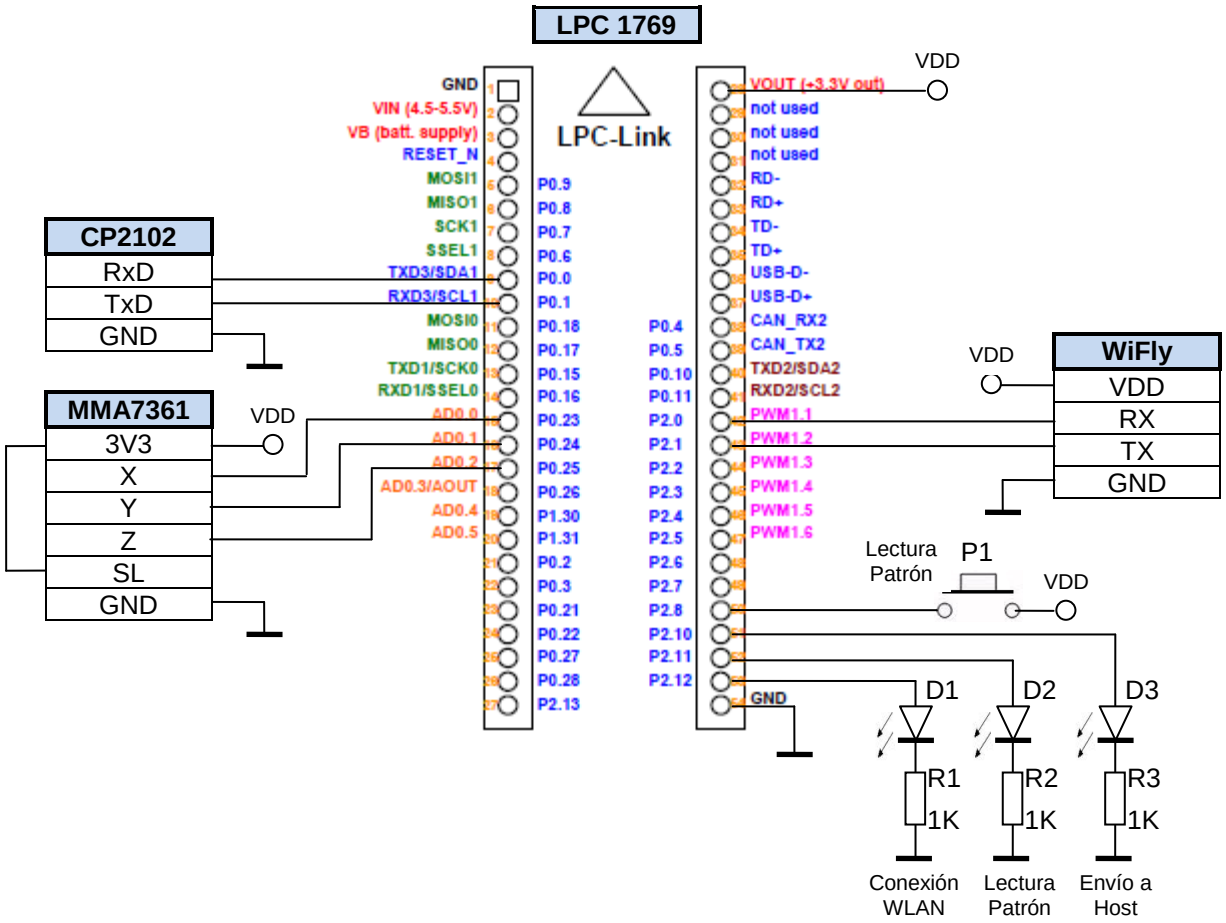



*Figura 6. Esquema de conexionado*

## 4.2 Módulos sistema embebido

El sistema está divido en varios módulos, cada uno de ellos con una funcionalidad concreta. Como base para una explicación detallada del funcionamiento del sistema detallaré cada uno de estos módulos.

### 4.2.1 Inicio del sistema

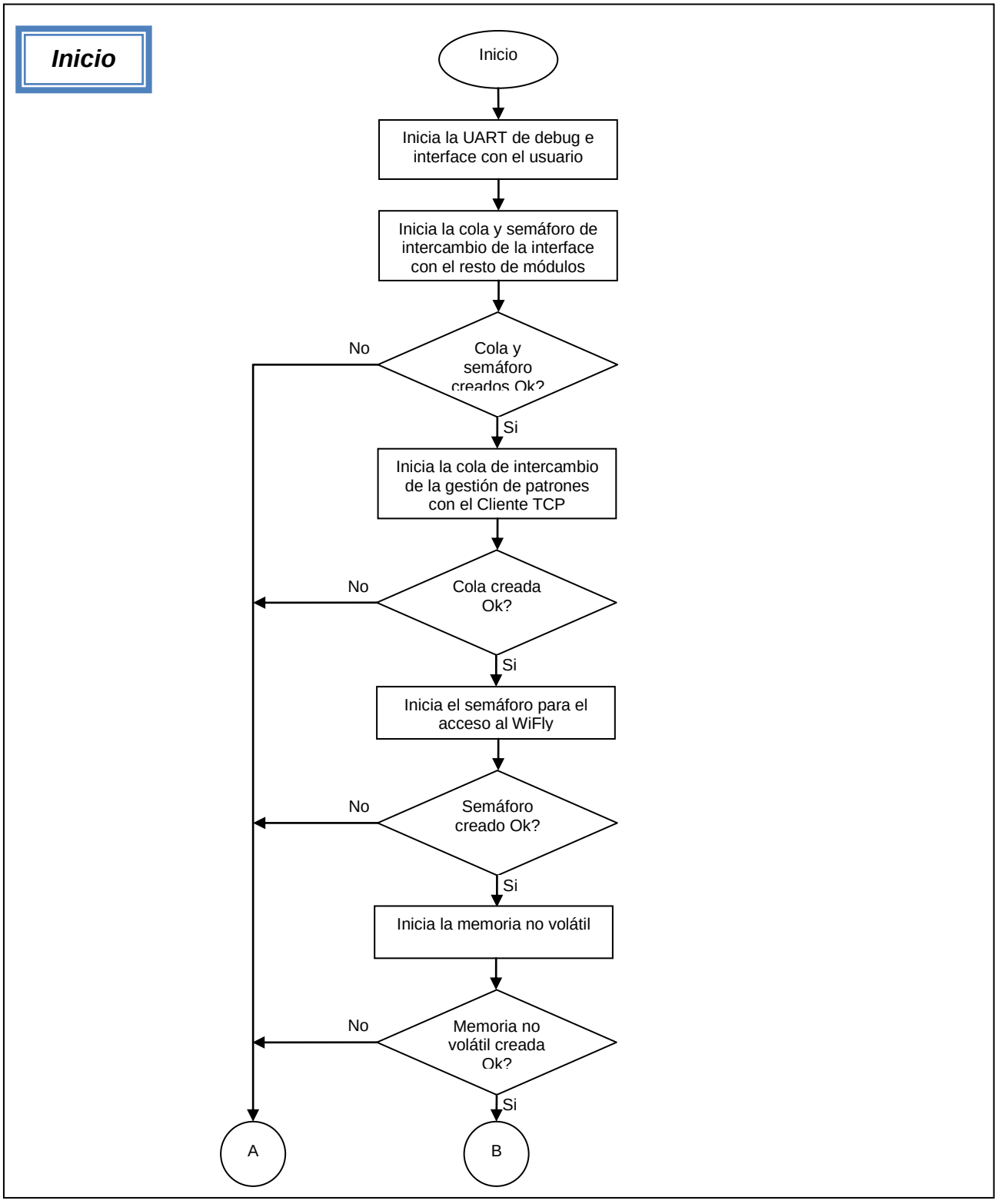



*Figura 7. Diagrama de flujo Inicio del sistema (1)*

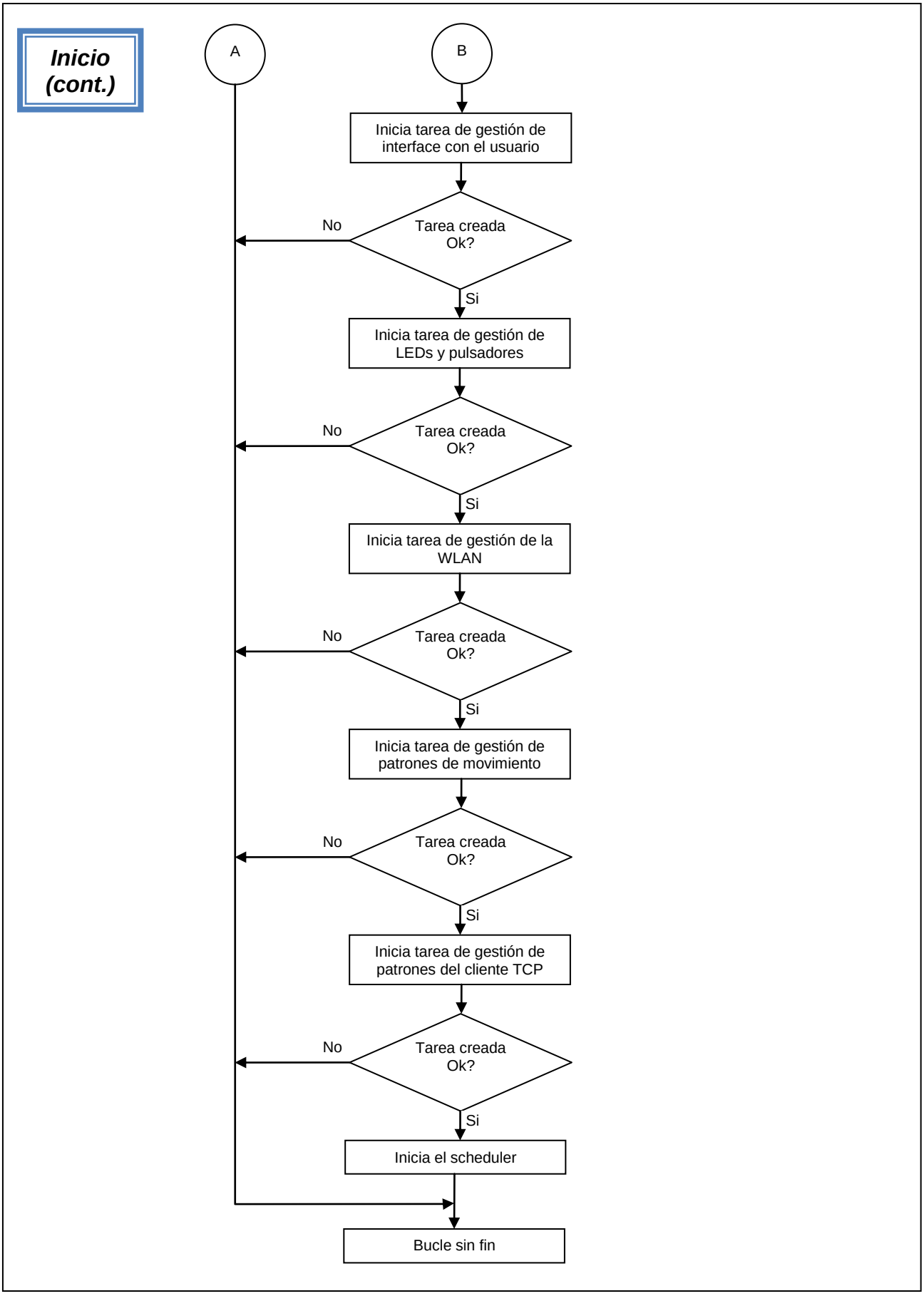



*Figura 8. Diagrama de flujo Inicio del sistema (2)*

Inicialmente se crean los recursos necesarios para la aplicación.

Si no es posible crear alguno de los recursos no se inicia el scheduler, quedando en un bucle infinito.

### 4.2.2 Tarea del interface con el usuario

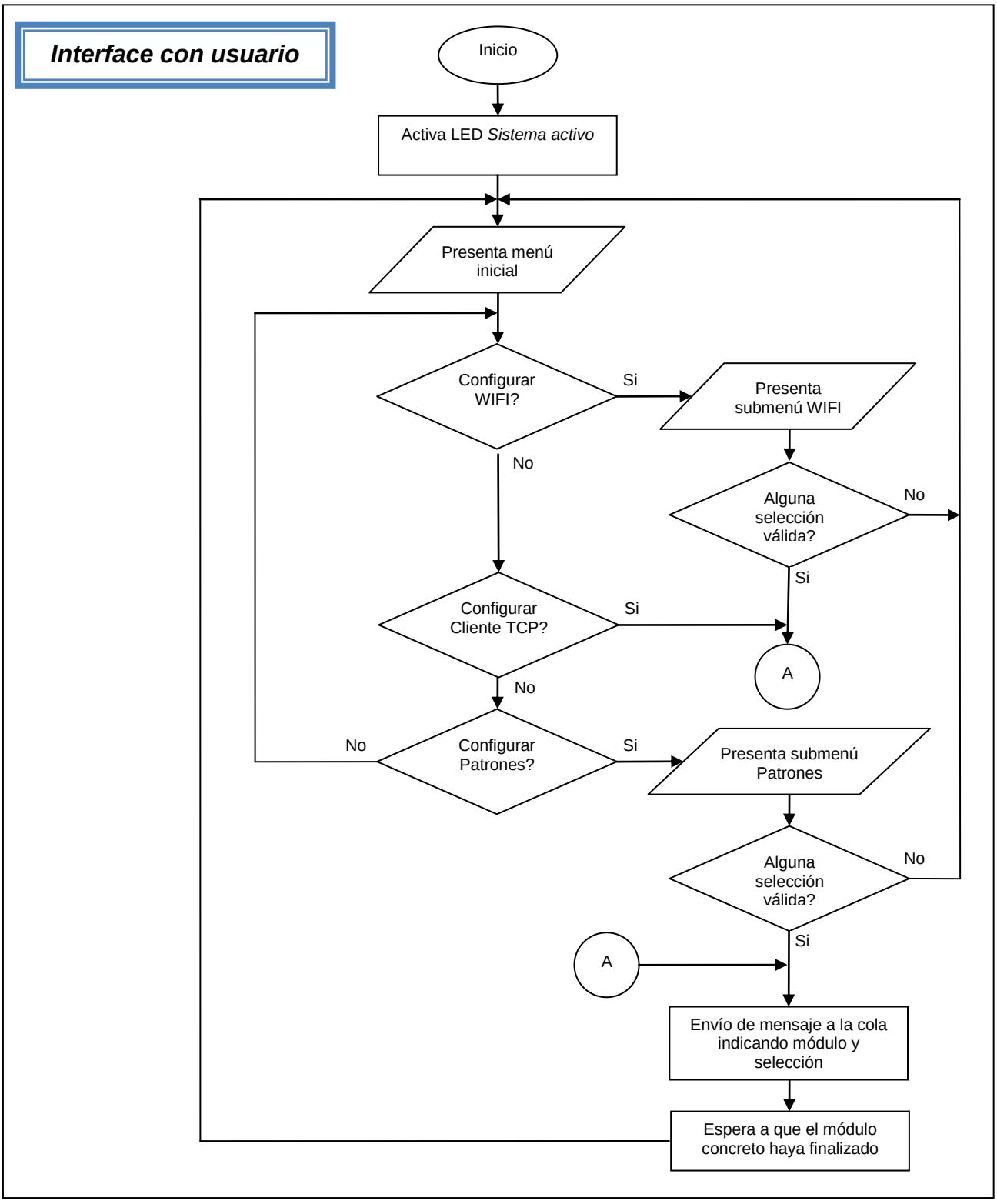



*Figura 9. Diagrama de flujo tarea interface con el usuario*

Si esta tarea es llamada es porque el sistema se ha inicializado correctamente, así como el scheduler, por lo que interpreta que todo ha sido correcto, activando el LED *Sistema activo*.

Para comunicar a los módulos la selección realizada utilizo una cola donde esta tarea escribe y de la que los demás módulos leen. Sólo si un módulo encuentra un mensaje para él lo sacará de la cola, en caso contrario los deja en ella.

El mensaje enviado tiene la siguiente estructura:

- *Módulo al que se le envía el mensaje*.
  Identifica a qué módulo va dirigido el mensaje.
- *Código del mensaje*.
  Indica la selección realizada en el menú o submenú.

Utiliza un semáforo de sincronismo, de forma que cuando envía el mensaje a la cola espera a que el módulo al que va dirigido lo devuelva, indicativo de que finalizado.

### 4.2.3 Tarea de gestión de la WLAN

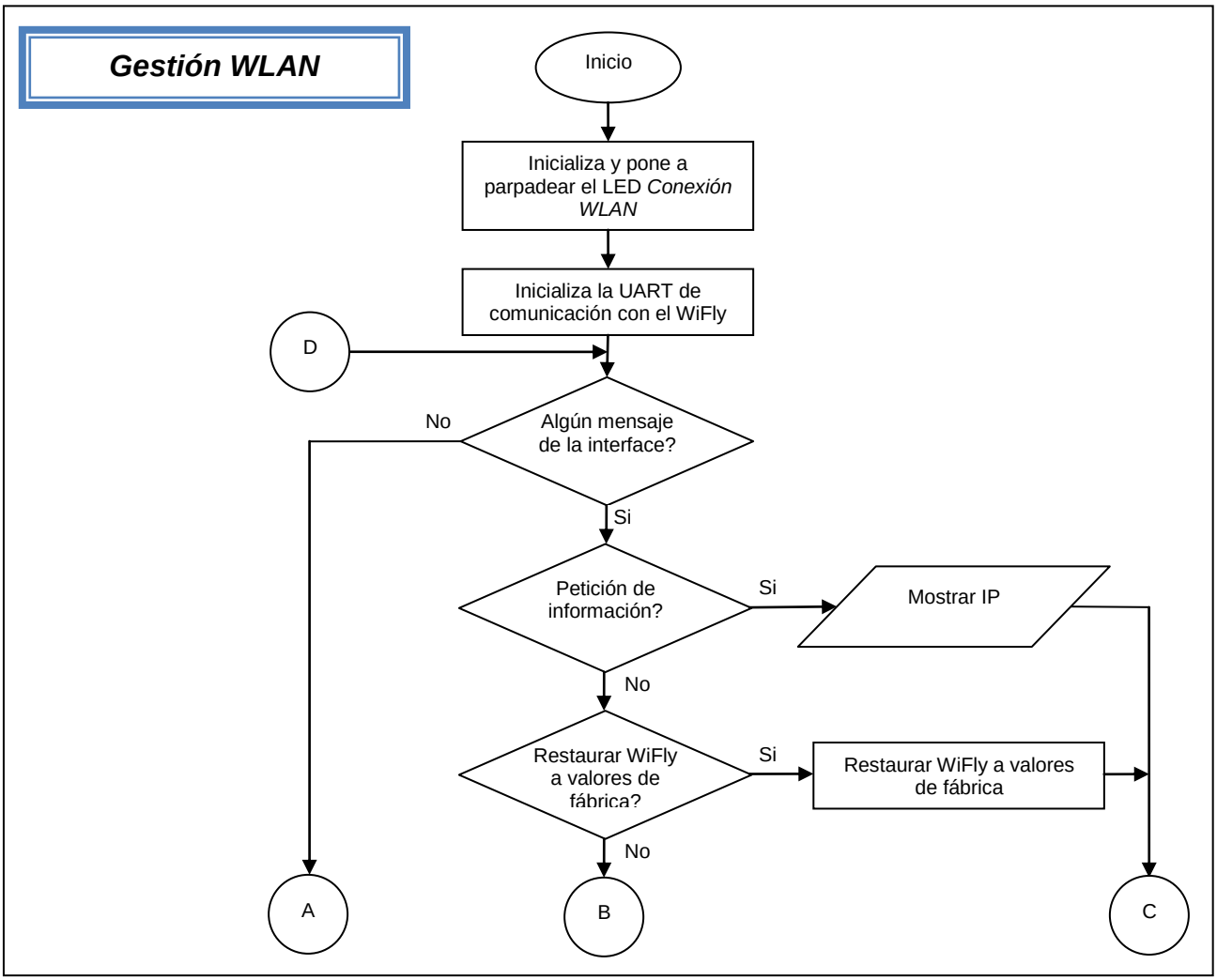



*Figura 10. Diagrama de flujo tarea de gestión de la WLAN (1)*

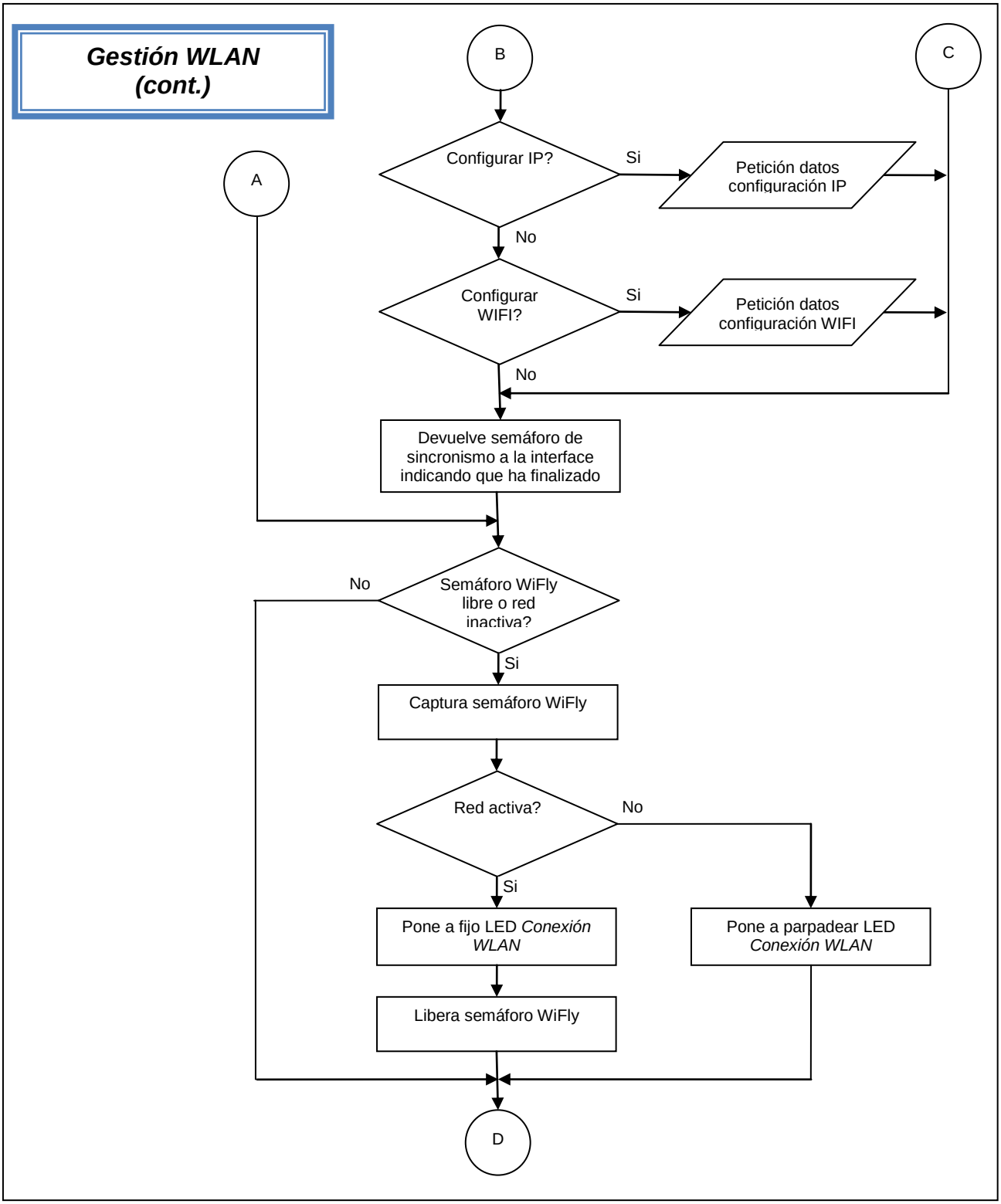



*Figura 11. Diagrama de flujo tarea de gestión de la WLAN (2)*

Constantemente vigila si tiene algún mensaje de la interface y cómo está el estado de la red.

En caso de detectar la red inactiva simplemente captura el semáforo de acceso al WiFly ya que es el propio módulo WiFly el que reconecta con la WLAN.

### 4.2.4 Tarea cliente TCP

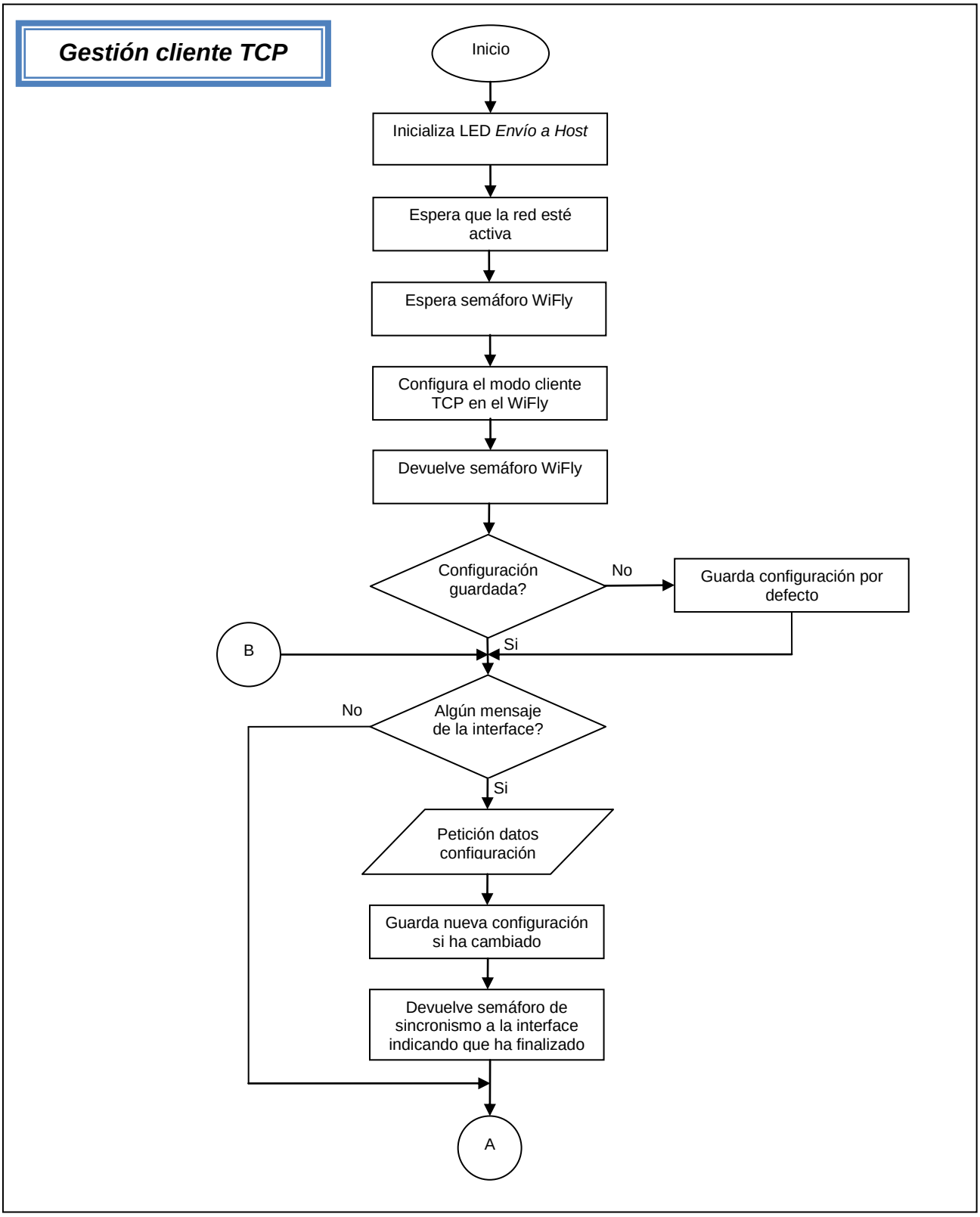



*Figura 12. Diagrama de flujo tarea de gestión del cliente TCP (1)*

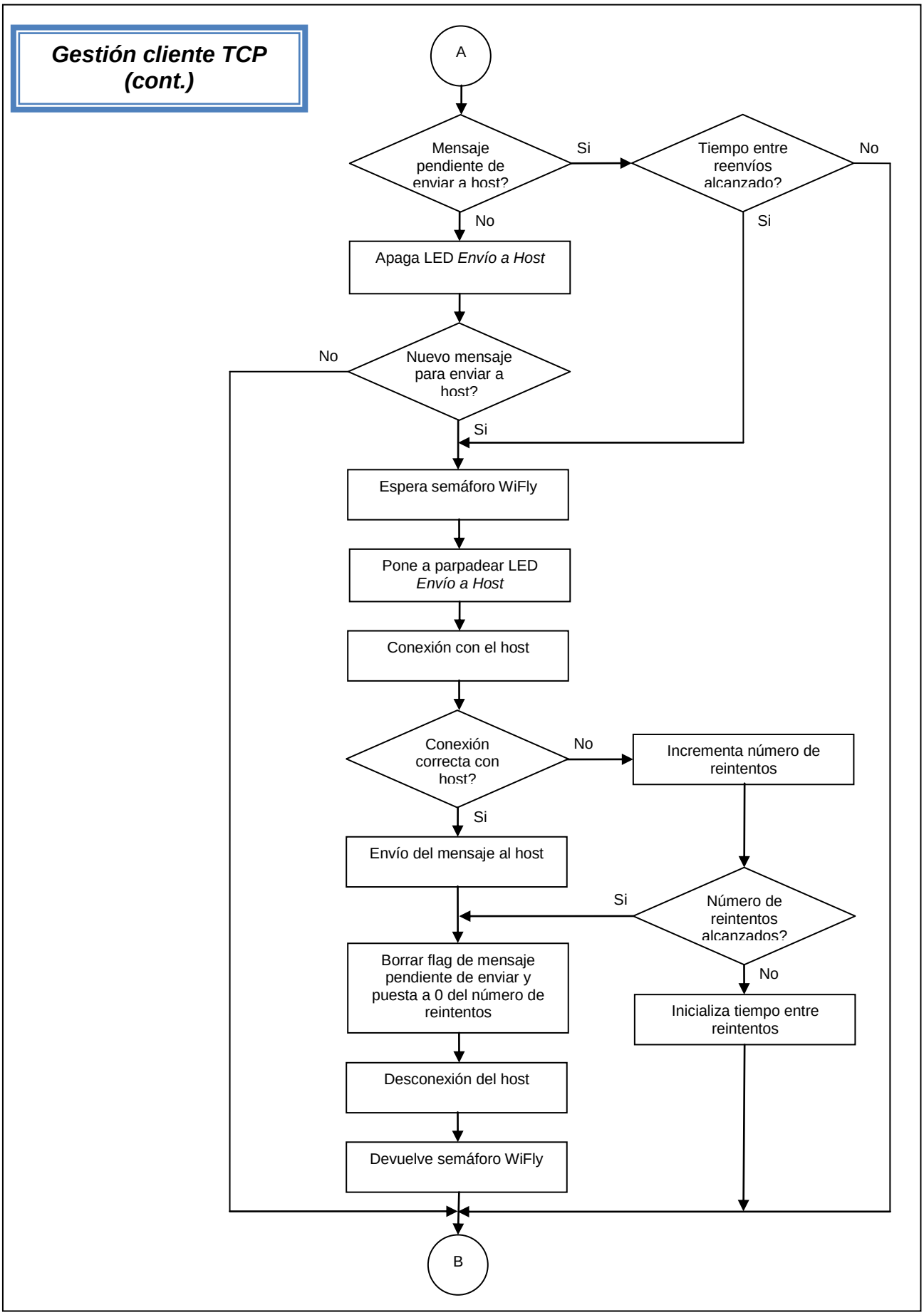



*Figura 13. Diagrama de flujo tarea de gestión del cliente TCP (2)*

Constantemente vigila si tiene algún mensaje de la interface y si tiene algún mensaje que enviar o reenviar al host.

El mensaje que envía al host es estrictamente el mismo que recibe de la gestión de patrones, no realiza ningún tratamiento con él.

Por defecto los parámetros del host son: IP 192.168.254.1, puerto 5123. Estos parámetros los guarda en la memoria no volátil, por lo que si no se le configuran otros utilizará los últimos configurados, o los ya comentados de defecto si no se han configurado otros.

Realiza hasta tres intentos de enviar el mensaje al host, con un tiempo entre reintentos de cinco segundos. Una vez realizados los reintentos, aunque no consiga conectar con el host, deshecha el mensaje pendiente.

### 4.2.5 Tarea de gestión de patrones

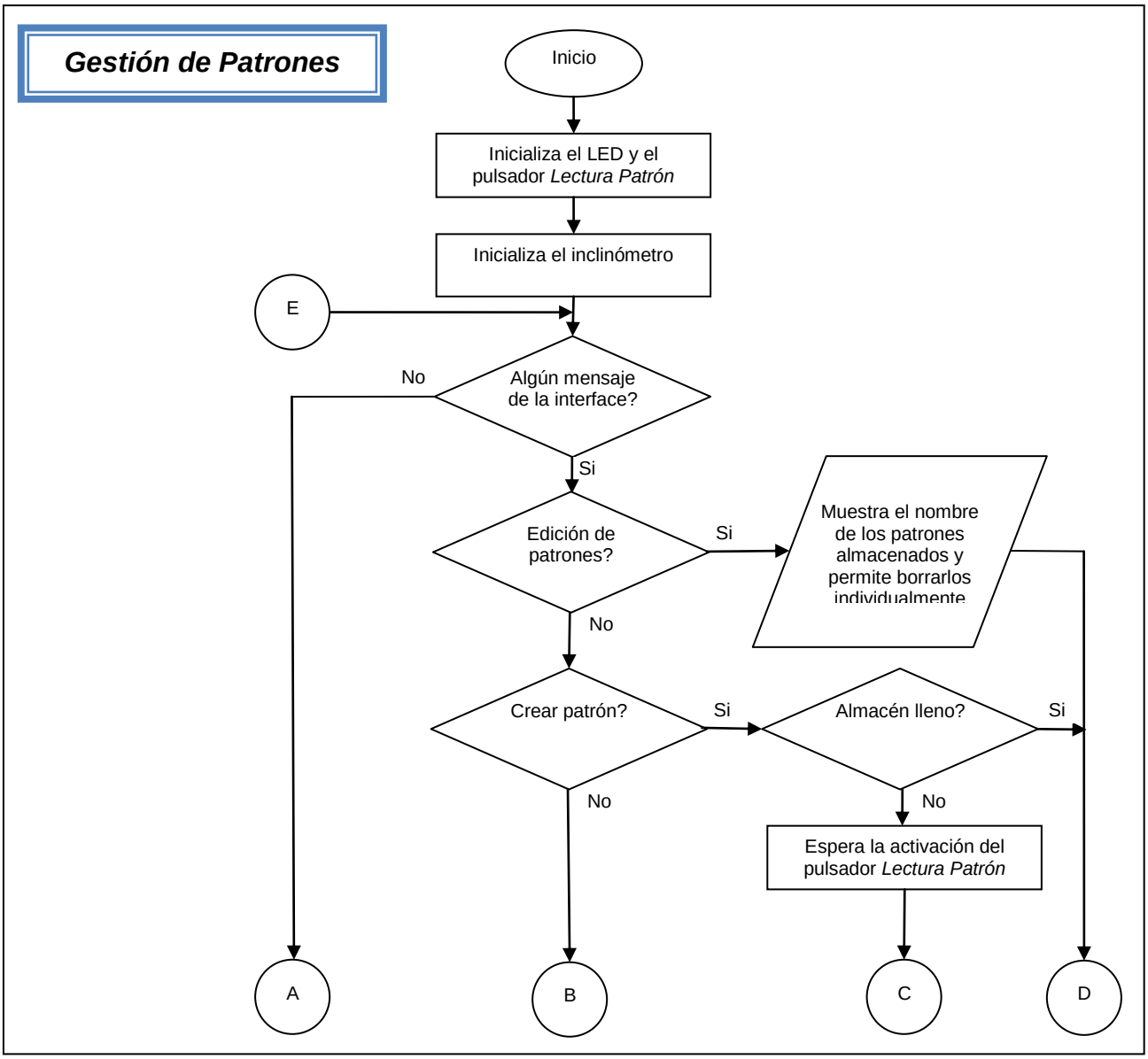



*Figura 14. Diagrama de flujo tarea de gestión de patrones (1)*

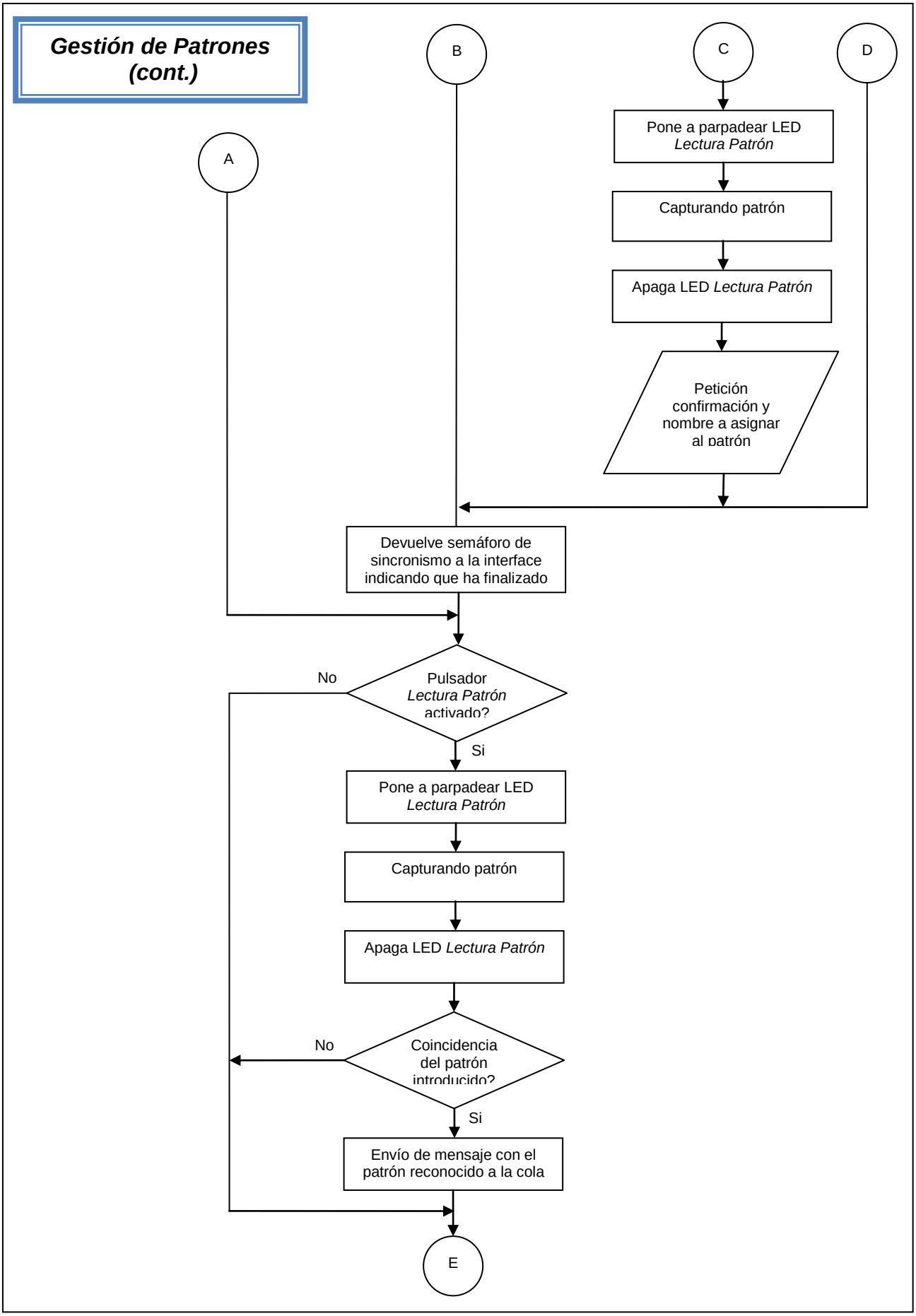



*Figura 15. Diagrama de flujo tarea de gestión de patrones (2)*

El sistema está limitado a un máximo de diez patrones. Este límite no tiene ninguna razón técnica, simplemente es porque para la demostración de funcionamiento son más que suficientes y en la edición de patrones es más cómodo con pocos.

Al guardar un patrón se le ha de asignar un nombre, que será su identificador. Este identificador será el que se muestre en la edición y el que se enviará en la cola para que sea reenviado al host. El tamaño máximo de este nombre es de diez caracteres, más el carácter nulo de terminación.

### 4.2.6 Tarea de gestión de LEDs y pulsadores

Para la gestión de LEDs y pulsadores apoyo en otra librería, la librería *digIO*, que permite gestionar hasta 32 entradas y 32 salida digitales del LPC 1769. Cada una de las posibles entradas/salidas se configura indicando el puerto y el pin a la que está conectado físicamente y un índice en el rango [0..31] para su referencia en el programa (índice independiente entre las entradas y las salidas).

Como especialización de esta tarea, pensada para LEDs y pulsadores, configura los LEDs sin ninguna resistencia pull-up/pull-down integradas en el microcontolador y la salida activa es en valor alto. Los pulsadores los configura como pull-down para que tenga una referencia a negativo por lo que al pulsador hay que conectarlo a un común positivo (VDD), de forma que al activarse lleve un valor alto a la patilla.

El hecho de crear una tarea para controlar los LEDs y pulsadores es para poder dotar a los módulos que lo precisen de dos ayudas:

- La tarea lee periódicamente las entradas que se le indiquen, realizando un polling cada 10 ms.
  A parte del polling, que no tendría mayor sentido facilita, un puntero a la posición de memoria que será actualizada con el valor del pulsador, esto permite tener un equivalente a una hook function pero con un simple puntero.
- Esta tarea permite asignar un periodo de parpadeo a un LED, siendo ella la que se encarga de mantenerlo, facilitando así la gestión de este tipo de comportamiento al resto de módulos. El periodo mínimo de parpadeo es de 20 ms, más que suficiente si se trata para LED indicativos.

### 4.2.7 Gestión de memoria no volátil

Para que el sistema no pierda los patrones por el hecho de quitarle la alimentación he querido dotarlo de memoria no volátil. Aprovechando este hecho también he guardado algún otro dato de configuración, concretamente la IP y el puerto del host con el que hay que conectar para enviarle el mensaje al encontrar una coincidencia con los patrones.

El resto de parámetros de configuración corresponden a la configuración de la WLAN y aprovecho que el módulo WiFly ya puede almacenarlos en su memoria no volátil.

En la gestión de la memoria no volátil tenía dos opciones:

- La memoria flash interna del propio microcontrolador.
- La memoria EEPROM del chip 24LC64 del que está dotada la placa del LPC1769.

Ambas soluciones eran válidas a nivel de capacidad de memoria, aunque con este microcontrolador con 512 KB de memoria flash y teniendo en cuenta que el software LPCXpresso gratuito sólo pueden programar los primeros 256 KB y no toda esa memoria está usada, quedan un mínimo de otros 256 KB de memoria flash utilizable como memoria no volátil.

La memoria del 24LC64 tiene la ventaja de soportar mayor cantidad de escrituras. De todas formas teniendo en cuenta que esta memoria no tengo pensada utilizarla como memoria temporal, sólo para guardar configuraciones y patrones, y que no preveo que esto datos cambien muy a menudo, no creo que esta limitación sea determinante.

Finalmente opté por utilizar la memoria flash interna del microcontrolador ya que pensando en un sistema real, el hecho de utilizarla evita tener que utilizar componentes externos, con lo que disminuye el coste, el tamaño y el consumo (siempre pensando en que no tuviera que utilizar un microcontrolador superior).

Como librería para la gestión de la memoria no volátil, utilizando la memoria flash del microcontrolador, he utilizado una proporcionada por el propio fabricante.

Esta librería minimiza tres problemas básicos que surgen al utilizar la memoria flash interna para almacenar variables:

- El borrado de la memoria es por sectores.
  Cuando la memoria está borrada el valor de sus posiciones es 0xFF, se programa escribiendo 0s, pero no se puede poner un 1 sin borrar el sector completo.
- La vida útil de la memoria.
  El hecho de tener que borrar la memoria cada vez que se quiera cambiar su valor provoca que haya que tener muy en cuenta el número de ciclos máximos que permite la memoria.
- La reprogramación de la ubicación de los datos.
  Si una variable varia muy a menudo provoca la reescritura de la misma zona de memoria hasta que esa zona falle mientras que habrá zona que apenas se habrán usado.

La idea de la librería es tener dos sectores de memoria, cada vez que haya que reescribir una variable se crea un nuevo registro en el sector y el anterior se ignora. Cuando el primer sector se llena, éste se copia al segundo y el primero es borrado, en este momento el sector válido será el segundo y cuando se llené se volverá a pasar al primero y así sucesivamente. De esta manera el número de reescrituras se distribuye por todo el sector y no se centra en una misma zona y en todo momento tenemos la información a salvo, por el hecho de que cuando borramos un sector tenemos la información en el otro.

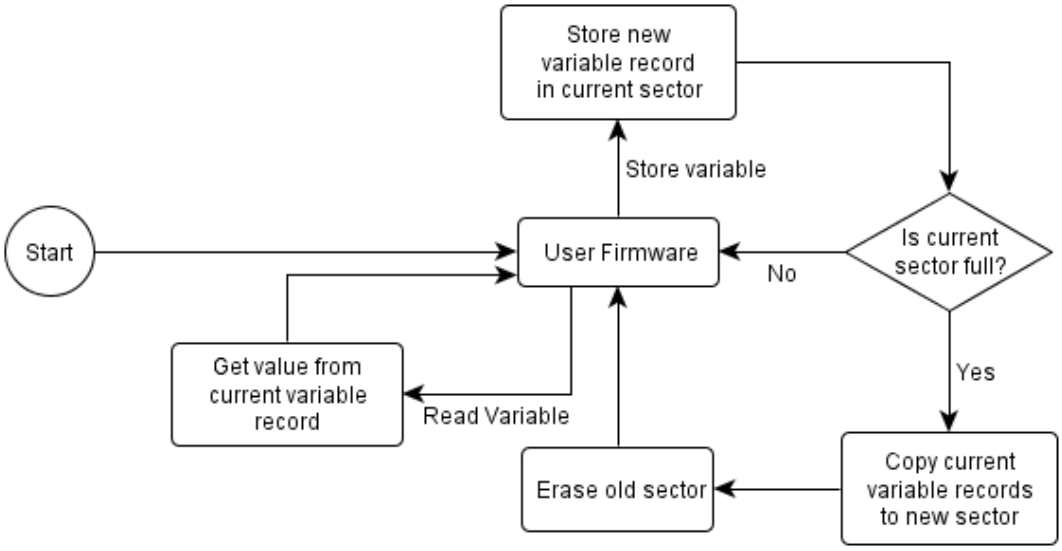



*Figura 16. Mecanismo básico de la gestión de la memoria flash interna*

Los sectores utilizados tienen una cabecera que indica su estado.

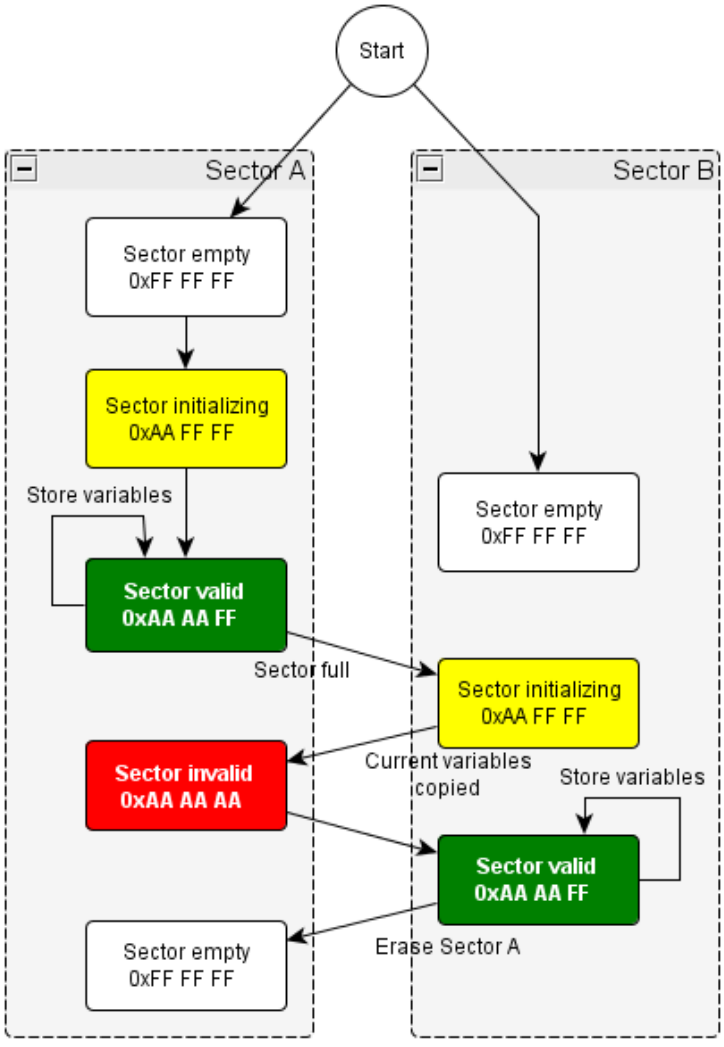



*Figura 17. Estado de los sectores en la gestión de la memoria flash interna*

Estos estados nos indican:

- Sector empty: El sector ha sido borrado y está libre para utilizarse.
- Sector inicializing: El sector está siendo inicializado para pasar a ser el válido y utilizarse para almacenar las variables, se está copiando la información del otro sector.
- Sector valid: El sector se está utilizando para almacenar las variables.
- Sector invalid: El sector se ha llenado y la información ya ha sido pasada al otro sector, este sector será borrado.

La cantidad y tamaño de las variables a almacenar tiene unas ciertas restricciones. Siendo:

- SECTOR_SIZE el tamaño de un sector utilizado en bytes
- MAX_VARIABLES el número máximo de variables
- MAX_VARIABLE_SIZE el tamaño máximo de una variable en bytes

Se ha de cumplir que:

- MAX_VARIABLES < ( (SECTOR_SIZE - 48) / (MAX_VARIABLE_SIZE + 4) )
- (MAX_VARIABLE_SIZE + 4) MOD 16 = 0
- 256 MOD (MAX_VARIABLE_SIZE + 4) = 0

Para este proyecto he utilizado los sectores 22 y 23, que empiezan en las direcciones 0x00040000 y 0x00048000 respectivamente. Estos sectores están encima de los 256KB que puede programar el LPCXpresso gratuito y cada uno es de 32 KB de capacidad, con lo que tengo 32 KB de capacidad de almacenaje de memoria no volátil menos 48 bytes que ocupa el estado del sector.

Aunque he definido un máximo de 20 variables, sólo utilizo 11 (10 patrones y una de los parámetros de configuración del host). El tamaño máximo de las variables que utilizo es de 60 bytes.

Con estos valore se puede apreciar que tengo la memoria no volátil infrautilizada.

## 4.3 Patrones de movimiento

### 4.3.1 Conceptos previos

Para poder interpretar los datos que un acelerómetro nos va a suministrar y poder genera información con dichos datos, primero hay que entender el concepto de aceleración.

Esta parte teórica es básica para poder determinar si el sistema es viable, con los elementos de que se dispone, y permite analizar el correcto funcionamiento del acelerómetro y del sistema de adquisición por los valores que leemos de él.

Un movimiento lineal, compuesto por un arranque y un paro (ir del punto A al B), va a generar lo mostrado en las siguientes gráficas:

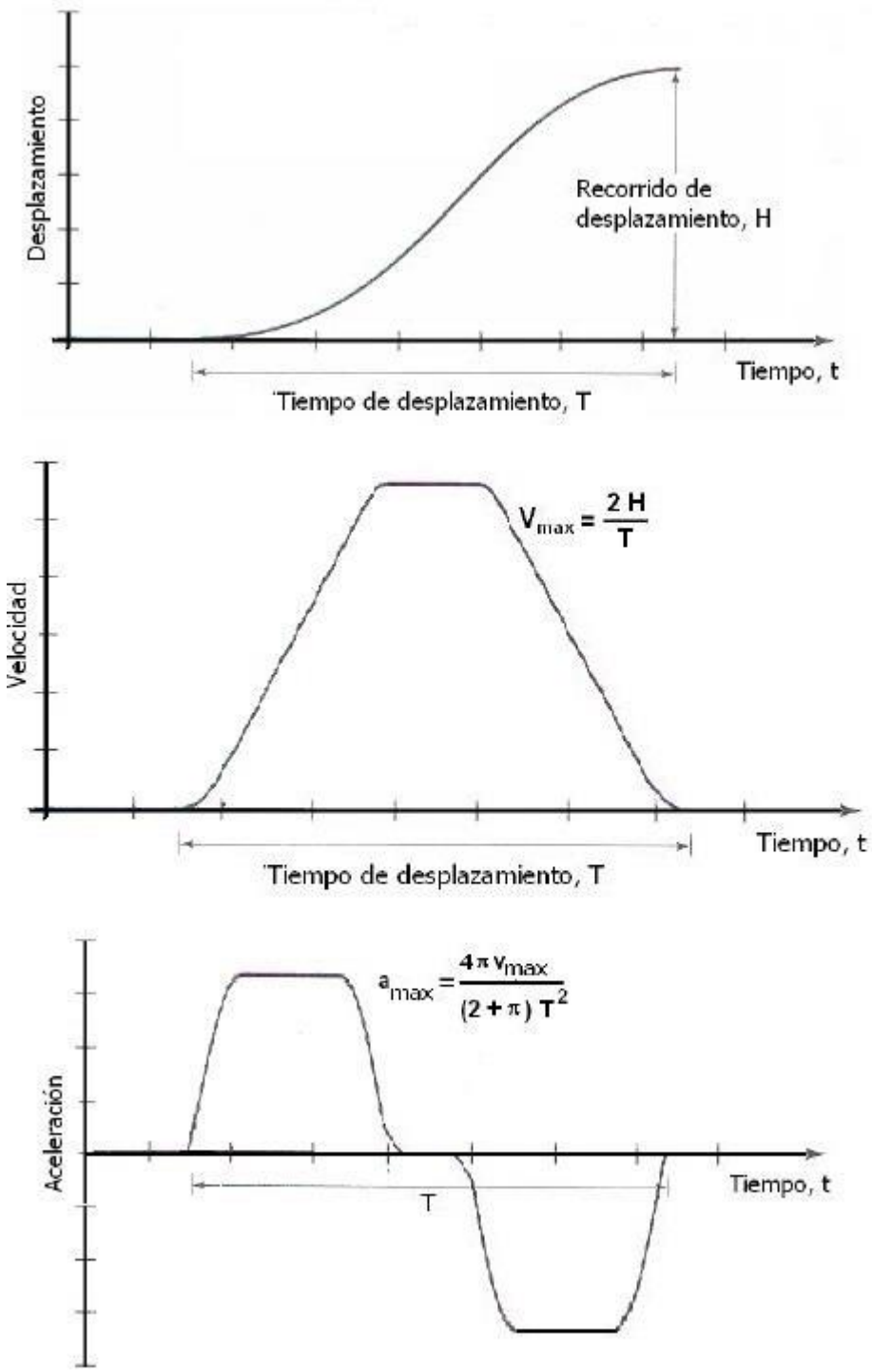



*Figura 18. Leyes de desplazamiento, velocidad y aceleración de un elemento con movimiento de traslación con arranque y parada*

Debido a las inercias, que todo cuerpo experimenta, nunca llegaremos a ver ángulos rectos en ninguna de las gráficas.

Centrándome en la gráfica de la aceleración, si el movimiento fuera de ida y vuelta (ir del punto A al B y sin parar en B volver al punto A), el resultado sería el siguiente:

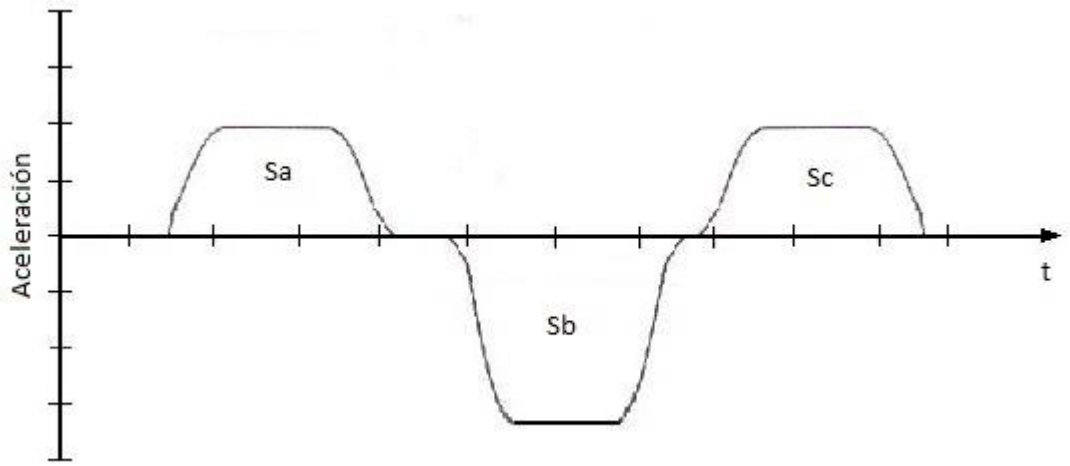



*Figura 19. Aceleración en un movimiento de ida y vuelta*

Donde, considerando que se pate de una velocidad de 0 m/s y se finaliza a la misma velocidad de 0 m/s, se cumpliría la relación entre las superficies: *Sa* + *Sc* = *Sb*.

Con este análisis de la previsión del comportamiento de la aceleración en un movimiento sobe un eje, en un movimiento simple (A→B) y en otro compuesto (A→B→A), es posible pensar en cómo almacenar movimientos en base a su aceleración asociada.

### 4.3.2 Identificación de un movimiento

Inicialmente sigo analizando los movimientos que afectan a un solo eje, por simplicidad, para finalmente extrapolarlo a varios ejes conjuntamente.

Las premisas que me marco para la gestión de patrones de movimiento es que el coste computacional y de almacenamiento de los patrones sea bajo.

Considerando que cada vez que un eje sufre un cambio de sentido experimenta un nuevo movimiento, parece sencillo pensar en la forma de detectarlos.

Una vez que se ha iniciado un movimiento no importa que incremente su aceleración mientras siga siendo en el mismo sentido, ya que lo único que indicaría es que se ha efectuado un movimiento conjunto más brusco, con cambios de aceleración, pero si continúa siendo en el mismo sentido sigue siendo parte del mismo movimiento.

Ya que lo que me interesa es capturar los cambios de sentido, para así determinar cuándo se produce un nuevo movimiento, sólo tengo que establecer qué considero que es un cambio de sentido.

Para determinar que se ha producido un movimiento establezco un umbral de detección. Si la aceleración supera este umbral y el sentido de movimiento es diferente al del último movimiento detectado considero que se ha producido un nuevo movimiento.

Teniendo en cuenta que se analiza el valor de la aceleración, la especificación de "*y el sentido de movimiento es diferente al del último movimiento detectado*" no significa que la aceleración baje del valor del umbral establecido, si no que sobrepase el valor del umbral pero con un valor inverso al último detectado

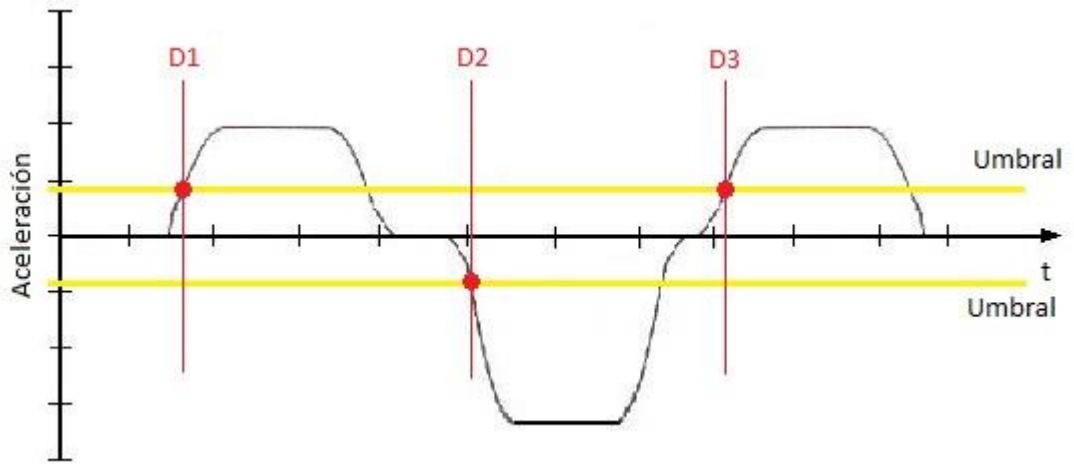



*Figura 20. Detección de movimientos*

En este gráfico se pueden ver los puntos, *D1*, *D2* y *D3*, en los que considero que se ha producido un nuevo movimiento. En estos puntos se ha llegado al valor del umbral establecido en sentido contrario al último movimiento detectado, exceptuando la primera detección (*D1*) en el que, al no haber ningún movimiento previo detectado, igualmente se considera que se ha producido un nuevo movimiento.

Con lo explicado hasta ahora se puede deducir fácilmente que en un movimiento simple (A→B) lo normal es que se detecten dos movimientos. Matizo lo de "*normal*", porque se podría llegar a dar el caso en que la desaceleración del movimiento fuera tan suave que el sistema no lo interpretaría como movimiento. Con esto se aprecia la relativa relevancia que tiene el valor del umbral, si es muy alto pueden perderse movimientos y si es muy bajo puede llegar a considerar movimientos a meras oscilaciones próximas al 0.

En el caso de trabajar con los tres ejes la detección de un movimiento es muy similar, aunque con algún matiz. Si los tres ejes se trataran independientemente se podría generar el problema de que movimientos que parecen idénticos sean interpretados como diferentes.

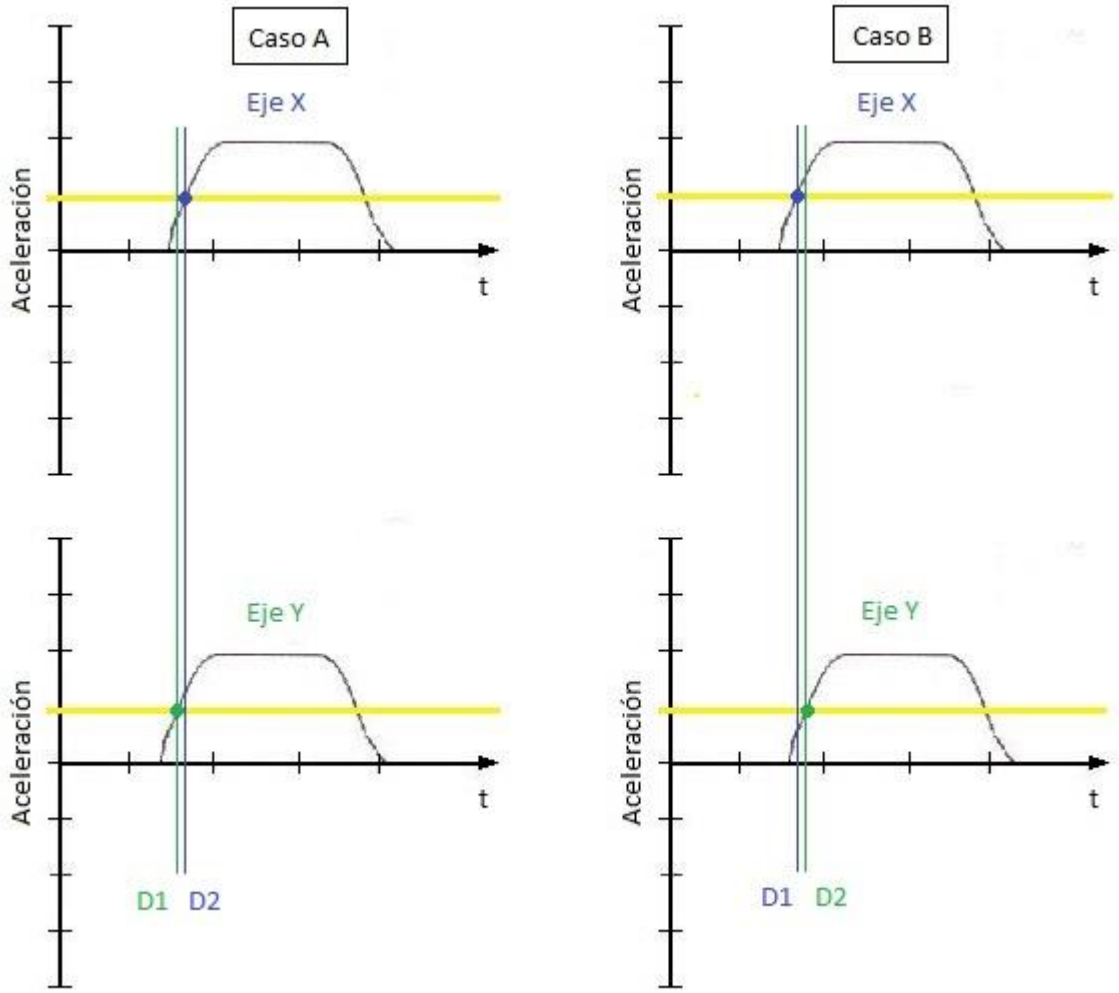



*Figura 21. Detección de movimientos en dos ejes en forma independiente*

En un movimiento en diagonal, en la que intervendrían dos ejes conjuntamente, podríamos tener cualquiera de los casos mostrados en estos últimos gráficos. La única diferencia entre un caso y el otro es que podría haber habido una pequeña inclinación diferente o el inicio del movimiento de un eje con respecto al otro podría haber variado un poco. Sin embargo en el caso A se detectaría que ha habido un movimiento en el Eje Y seguido de otro en el Eje X y en el caso B la secuencia de detección sería la inversa.

Para evitar esto la comparación entre un movimiento y otro la realizo teniendo en cuenta todos los ejes, considerando cada movimiento como un vector tridimensional. Utilizo el propio umbral para considerar si la diferencia de valor del eje que no ha llegado al umbral se puede considerar que corresponde al mismo vector o no. Cada vez que un eje cumple con las condiciones necesarias para considerar que ha habido un nuevo movimiento guarda el valor de todos los ejes (considerado como vector), este valor guardado será el siguiente que se compare con los nuevos valores.

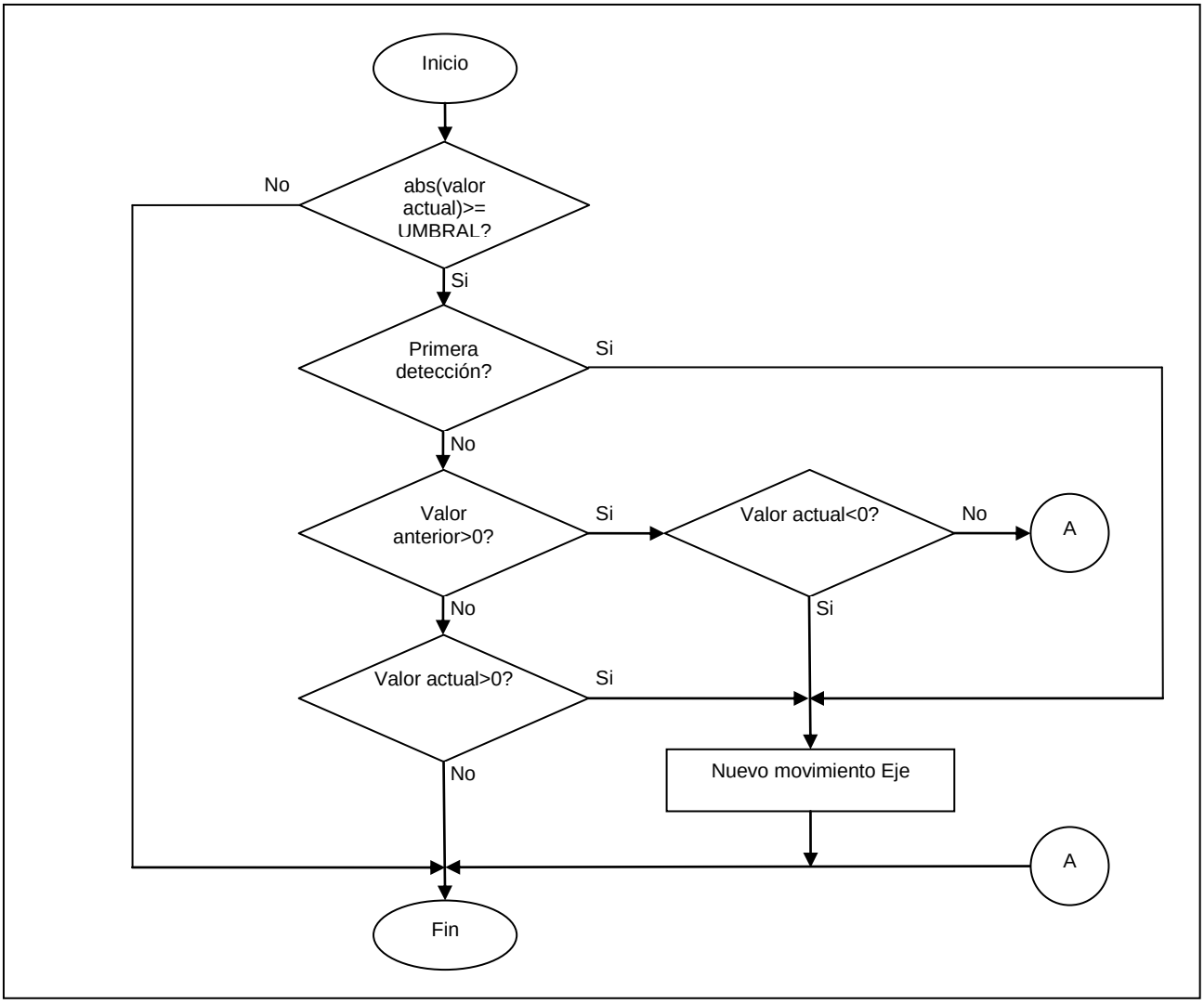



*Figura 22. Detección de nuevo movimiento en un eje*

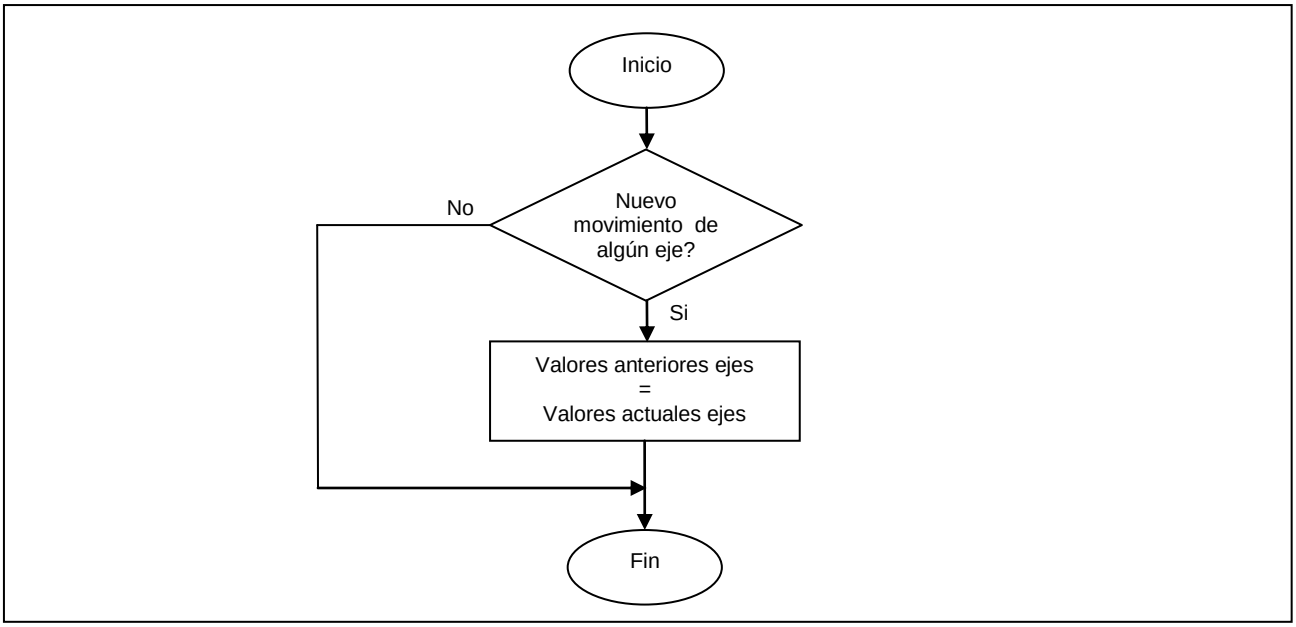



*Figura 23. Detección global de nuevo movimiento*

En estos diagramas de flujo de muestra el algoritmo utilizado para la detección de un movimiento. Si se detecta movimiento en un eje se guarda el valor actual de todos, con lo que la siguiente comprobación se realizará con esto últimos valores guardados.

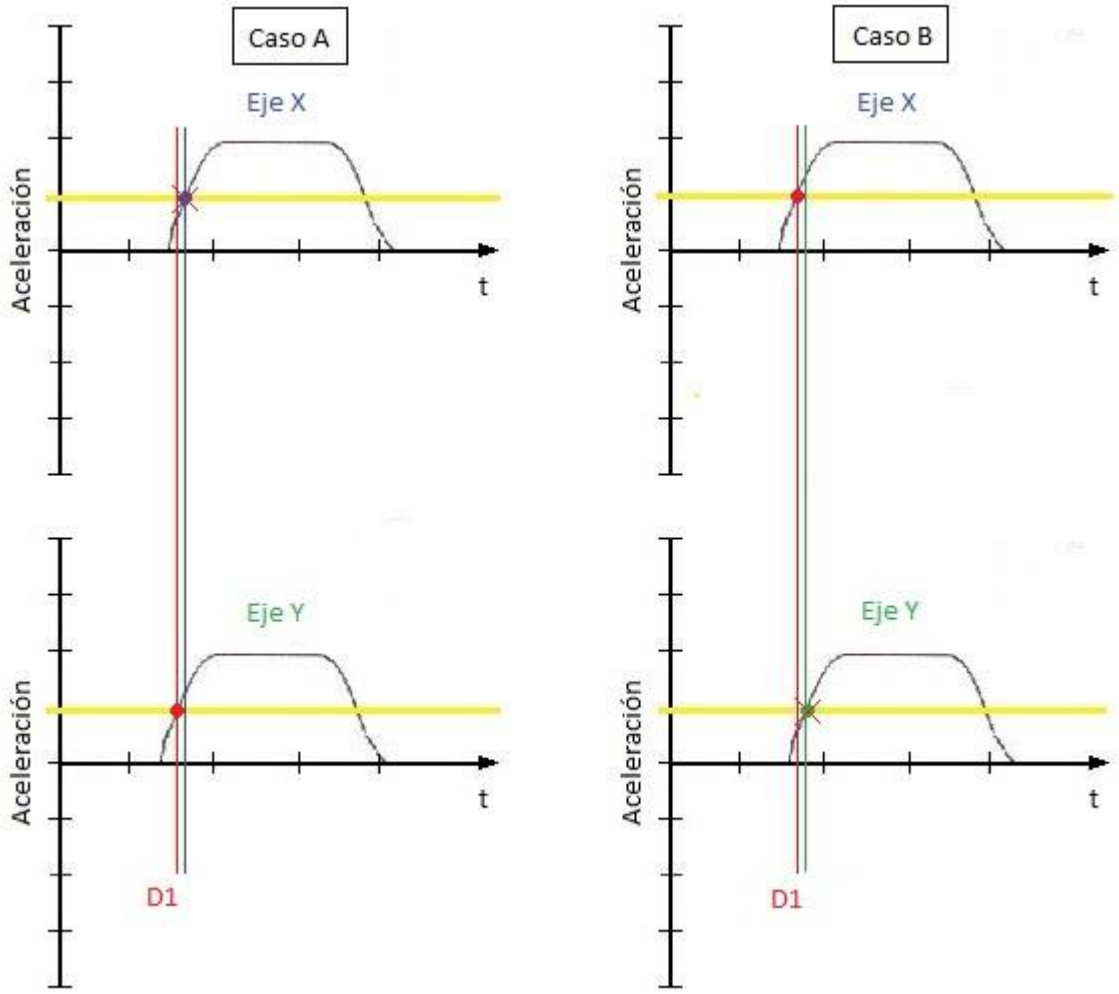



*Figura 24. Detección de movimientos en dos ejes en forma conjunta, consideración de un solo movimiento*

De esta forma cuando detecta un cambio de movimiento en el primer eje guarda también los valores del segundo, con lo que no considera que este varíe aunque sobrepase el umbral. Considerará que el valor del vector a guardar corresponde a los valores de los ejes en el momento en que el primer eje ha sobrepasado el umbral.

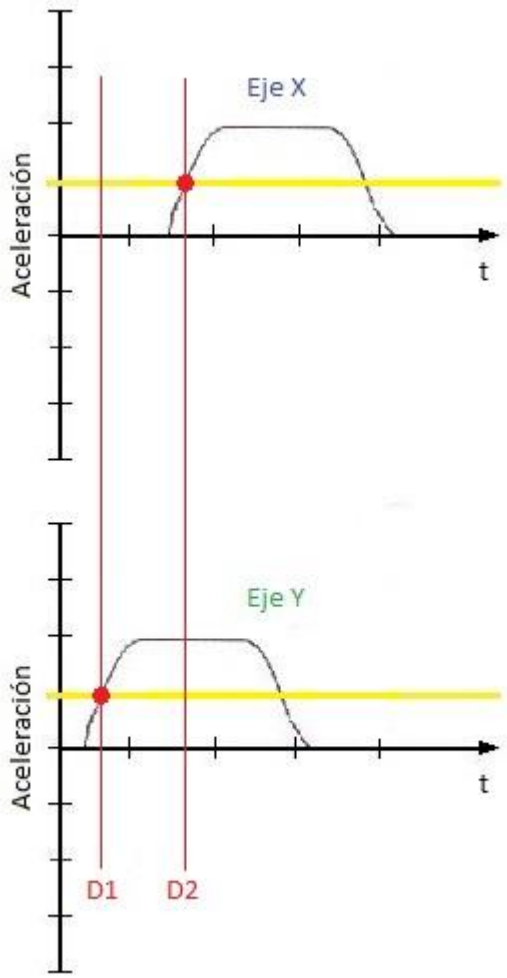



*Figura 25. Detección de movimientos en dos ejes en forma conjunta, consideración de dos movimientos*

En estas últimas gráficas se observa la detección de dos movimientos, cada uno de ellos provocado por uno de los ejes.

### 4.3.3 Estructura de un patrón

A modo de introducción, la estructura de un patrón está compuesta por una descripción y un número de vectores y un vector es un array con el valor que tenían los tres ejes en el momento de detectar un nuevo movimiento.

Cuando se están generando los patrones trabajo con los valores de los ejes en formato real, para tener más precisión en los cálculos, pero para su almacenamiento escalo el valor de cada eje al tamaño de un byte, ya que para esta aplicación no veo la necesidad de mayor precisión y permite que el espacio de almacenamiento necesario sea mucho más reducido.

| Estructura Vector | |
|---|---|
| Descripción campo | Tamaño campo |
| Valor eje X | 1 byte |
| Valor eje Y | 1 byte |
| Valor eje Z | 1 byte |

| Estructura Patrón | |
|---|---|
| Descripción campo | Tamaño campo |
| Descripción | 11 bytes |
| Nº vectores | 1 byte |
| Vectores | 16 Vector (48 bytes) |

*Figura 26. Estructura de un patrón de movimiento*

Un patrón contiene su descripción, el número de vectores que lo componen y el valor de dichos vectores. El número máximo de vectores es de 16.

Tal y como se explicó anteriormente un movimiento simple ocupa normalmente 2 vectores, ya que habría un vector por el movimiento de arranque y otro por el de paro.

El tamaño de un patrón es de 60 bytes, con lo que el espacio de almacenamiento necesario es muy reducido

### 4.3.4 Adaptación de las señales del acelerómetro

Para la creación real de los patrones algo fundamental es la lectura del valor de los ejes del acelerómetro.

Cuando se inicia el algoritmo de creación de patrones, escaneo de forma constante el valor de los ejes para detectar los cambios de movimiento. Este escaneo consiste en la digitalización de las entradas analógicas a las que están conectados los ejes.

Por la forma comentada en los puntos anteriores, la frecuencia de muestreo utilizada en la lectura de los canales analógicos no tiene relación con la cantidad de vectores que generará para el patrón que se está generando.

En las pruebas de lectura he detectado que tengo algunas lecturas muy fuera de rango, posiblemente por problemas en falsos contactos en las conexiones eléctricas entre el acelerómetro y el LPC cuando lo muevo o interferencias electromagnéticas. Por este motivo realiza una serie de tratamiento a las señales leídas de los ejes del acelerómetro.

El tratamiento que realizo se basa en dos filtros:

- **Filtro de mediana**
  De cada tres lecturas sólo tomo la mediana como valor correcto. Esto me filtra algunos valores que alcanzan valores extremos esporádicamente.

- **Filtro paso bajo IIR de 1$^{er}$ orden**

  Aún con el primer filtro, la señal oscila demasiado, por lo que realizo un filtro paso bajo de primer orden. Utilizo el algoritmo de un IIR (*Infinite Impulse Response* o Respuesta infinita al impulso) porque no necesita un orden alto para conseguir los resultados deseados y computacionalmente es muy sencillo y rápido.

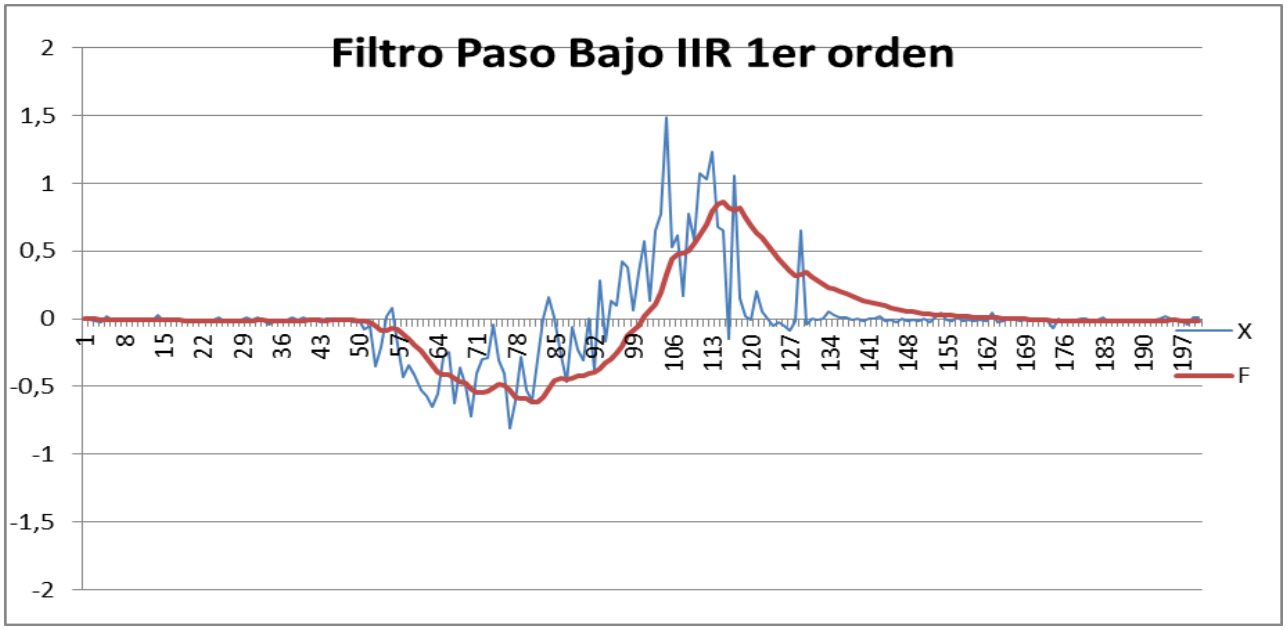



*Figura 27. Filtro tratamiento señal acelerómetro*

En esta figura se puede apreciar la señal procedente de un eje del acelerómetro (*X*) y el valor en la salida del filtro (*F*).

Estos filtros son realizados en los tres ejes y el valor resultante de ellos es el que utilizo en el resto del programa.

Hay que tener en cuenta que el acelerómetro realmente transmite las fuerzas G que le afectan, por lo que si está quieto indicará que está recibiendo una fuerza de 1 G en sentido perpendicular al suelo. Como lo que interesa es obtener la aceleración del movimiento que se realiza hay que eliminar la aceleración que percibe de la gravedad terrestre. Para compensar este hecho lo que hago es que a la hora de empezar a capturar un patrón, guardo los valores iniciales como valores de offset. Este valor lo resto de todas las lecturas que realizo mientras estoy capturando el patrón.

### 4.3.5 Comparación de patrones

Para la comparación de patrones utilizo un valor de margen de comparación de valores de los vectores.

Para considerar iguales dos parones se ha de cumplir que:

- Tengan el mismo número de vectores.
- En cada vector, la diferencia entre cada uno de los componentes de un patrón con respecto al otro ha de ser menor o igual al margen de comparación:

```
( abs(patron1.vector[i].x - patron2.vector[i].x) <= margen ) &&
( abs(patron1.vector[i].y - patron2.vector[i].y) <= margen ) &&
( abs(patron1.vector[i].z - patron2.vector[i].z) <= margen )
```

# 5 Conclusiones

## 5.1 Propuesta de mejoras

Una vez desarrollado y probado el proyecto, hay una serie de mejora que creo que se podrían aplicar:

- **Tamaño del gadget**
  Si ha de ser manipulado de una forma cómoda tendría que tener un tamaño manejable. Esto se podría lograr si se desarrollara un sistema real, ya que tiene menos requerimientos que un teléfono móvil.
- **Tipo de acelerómetro**
  Si el gadget se construye de forma compacta, con el microcontrolador y acelerómetro en la misma placa, no habría problema con utilizar este acelerómetro. Sin embargo si el acelerómetro es una extensión conectada con cables habría que utilizar otro tipo, ya que las salidas analógicas de tensión (y más a estas bajas tensiones) son muy poco inmunes a los ruidos y es un tipo de señal pensado para estar a muy corta distancia el emisor del receptor. Hay acelerómetros que transmiten los datos por medio de una conexión digital, SPI o $I^2C$, que además necesitan menos cables.
- **Envío de mensajes al host**
  Este sistema envía el mensaje al host de una forma plana, tal y como es, si se utilizara en un sistema de control de acceso habría que encriptar la información para que no fuera tan vulnerable.
- **Interacción con el sistema**
  Se podría utilizar la conexión WIFI con el equipo para su configuración. Dotándolo de un sencillo servidor web y con una configuración WLAN por defecto para poder conectarse a él. Sobraría el convertidor USB-UART, simplificando y abaratando el sistema.
- **Detección de movimientos**
  Habría que trabajar más en la detección de movimientos ya que a veces resulta demasiado "selectivo", ya que hacemos movimientos de los que no somos conscientes. También había que analizar si interesa que el sistema considere iguales movimientos realizados sosteniendo al acelerómetro de formas muy diferentes (cambio de base del sistema de vectores).

## 5.2 Autoevaluación

En la realización de este proyecto he podido profundizar en el mundo de los sistemas embebidos actuales. Aunque ya había tenido algún contacto con los microcontroladores, no había sido con sistemas tan completos y las aplicaciones eran más sencillas.

El llegar a este punto no ha sido fácil, no por la complejidad de estos sistemas, por la compresión en el tiempo. Realmente ha habido varios subsistemas a estudiar a los que me hubiera gustado dedicar más tiempo.

Como en todo proyecto, siempre hay algunas partes que mejorarías o en las que harías más hincapié, aunque en general estoy contento con este.

Cuando he creado librerías no las he hecho única y exclusivamente pensando en la PAC a la que iba destinada, las he hecho pensando en qué debería tener para estar completa sin importar dónde se utilizara, aunque por tiempo hay algunas que no he acabado de completar tal y como me gustaría. Este es un hecho que me ha hecho "perder" más tiempo del necesario y lo asumo.

En resumen aunque la cantidad de horas dedicadas ha sido bastante elevada, y sobre todo si se compara con los créditos que tiene asignados, lo he pasado bien realizándolo.

# 6 Glosario

**ADC** Analog to Digital Converter (Conversor Analógico Digital), dispositivo electrónico capaz de convertir una entrada analógica de voltaje en un valor binario.

**DNS** Domain Name System (Sistema de Nombres de Dominio), sistema de nomenclatura jerárquica para computadoras, servicios o cualquier recurso conectado a Internet o a una red privada.

**EEPROM** Electrically Erasable Programmable Read-Only Memory (Memoria de Solo Lectura Programable y Borrable Electricamente), tipo de memoria ROM que puede ser programada, borrada y reprogramada eléctricamente.

**IIR** Infinite Impulse Response (Respuesta infinita al impulso), tipo de filtros digitales en el que, como su nombre indica, si la entrada es una señal impulso, la salida tendrá un número infinito de términos no nulos, es decir, nunca vuelve al reposo.

**IP** Internet Protocol (Protocolo de Internet), estándar que se emplea para el envío y recepción de información mediante una red

**LAN** Local Area Network (Red de Área Local), red que conecta los ordenadores en un área relativamente pequeña y predeterminada.

**LED** Light Emitting Diode (Diodo Emisor de Luz), componente electrónico cuya función principal es convertir la energía eléctrica en una fuente luminosa.

**SSID** Service Set IDentifier (Identificador de Conjunto de Servicio), identificador de red inalámbrica.

**UART** Universal Asynchronous Receiver-Transmitter (Transmisor-Receptor Asíncrono Universal), controla los puertos y dispositivos serie.

**USB** Universal Serial Bus (Bus Universal en Serie), permite conectar diversos periféricos.

**WIFI** Wireless Fidelity (Fidelidad Inalámbrica), sistema de comunicación de datos inalámbrico.

**WLAN** Wireless Local Area Network (Red de Área Local Inalámbrica), sistema de comunicación de datos inalámbrico.

# 7 Referencias y bibliografía

*Curso Cálculo de componentes de transmisiones mecánicas* [en línea]. Gonzalo González Rey, Mayo 2009.
<http://www.mailxmail.com/curso-calculo-componentes-transmisiones-mecanicas/transmisiones-mecanicas-concepto>
[Consulta: 29 abril 2014].

FreeRTOS
<http://www.freertos.org/>
[Consulta: 31 marzo 2014].

Hercules SETUP utility
<http://www.hw-group.com/products/hercules/index_en.html>
[Consulta: 1 abril 2014].

*Introducción al filtrado digital*. Emilia Gómez Gutiérrez, noviembre 2009.
[Consulta: 11 mayo 2014].

LPCware
<http://www.lpcware.com/>
[Consulta: 31 marzo 2014].

Magus 3D gestos
<https://play.google.com/store/apps/details?id=com.stmp.magus&hl=es>
[Consulta: 15 junio 2014].

NXP Semiconductors
<http://www.nxp.com/>
[Consulta: 31 marzo 2014].

PlayStation Move
<http://store.sony.cl/w/playstation/move/>
[Consulta: 15 junio 2014].

PuTTY
<http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html>
[Consulta: 1 abril 2014].

Wii Remote Plus
<http://www.nintendo.com/wiiu/accessories/>
[Consulta: 15 junio 2014].

Wiki de la asignatura
<http://cv.uoc.edu/webapps/xwiki/wiki/matembeddedsystemslabhome/view/Main/WebHome>
[Consulta: 31 marzo 2014].

*VMware*. Máquinas virtuales.
<http://www.vmware.com/es>
[Consulta: 26 febrero 2014].

# 8 Anexos

## 8.1 Manual de usuario

### 8.1.1 Requisitos iniciales

Para poner el sistema en servicio hay que cumplir los siguientes requisitos:

- La UART del módulo WiFly esté configurado para comunicarse a 9600 baudios, 8 bits de datos, sin paridad y 1 bit de stop.
- El PC con el terminal al que se conecta el convertidor USB-UART CP2102 ha de tener instalado su driver correspondiente.
- La configuración del puerto serie a la que se ha conectado el CP2102 ha de ser 57600 baudios, 8 bits de datos, sin paridad y 1 bit de stop.

### 8.1.2 Configuración

Al alimentar el sistema el terminal nos muestra el siguiente menú:

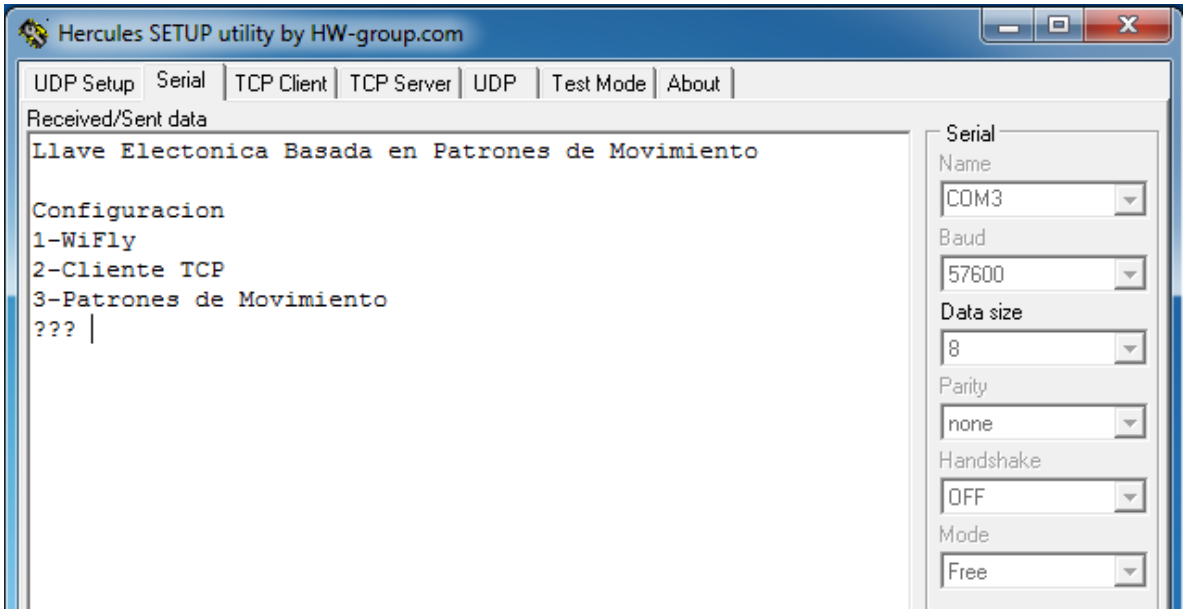



*Figura 28. Menú general*

A través de este menú general podemos navegar por las diversas opciones, seleccionando el número indicado accederemos a los parámetros concretos.

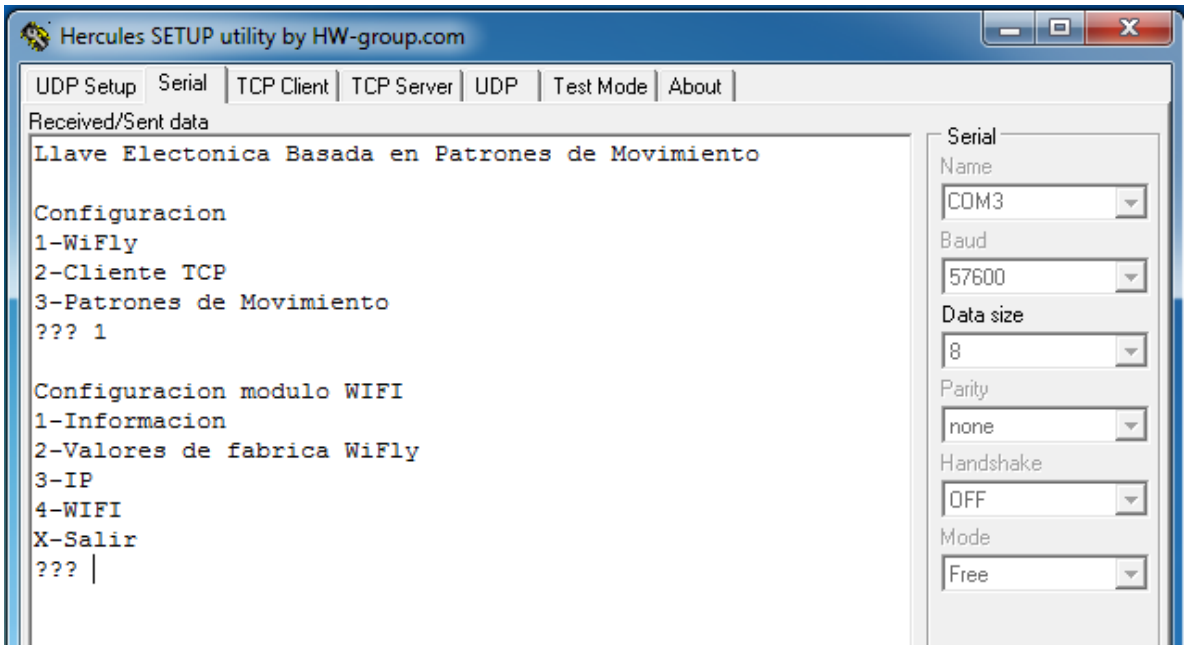



*Figura 29. Menú configuración WIFI*

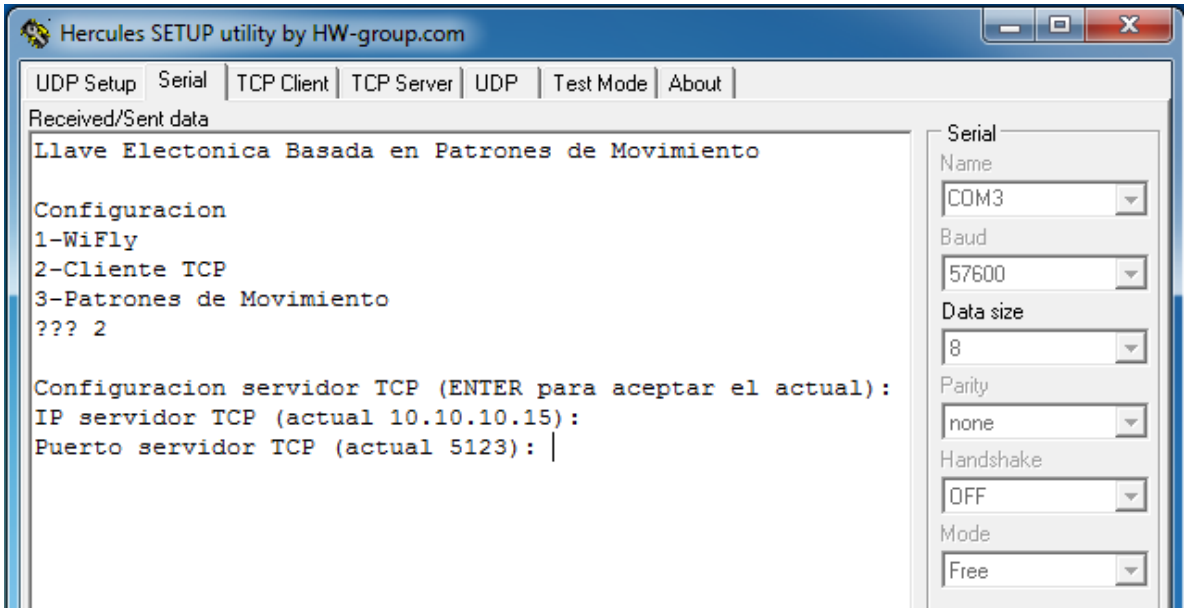



*Figura 30. Menú configuración Cliente TCP*

### 8.1.3 Edición de patrones

Para acceder a la edición de patrones hay que seleccionarlo en el menú general:

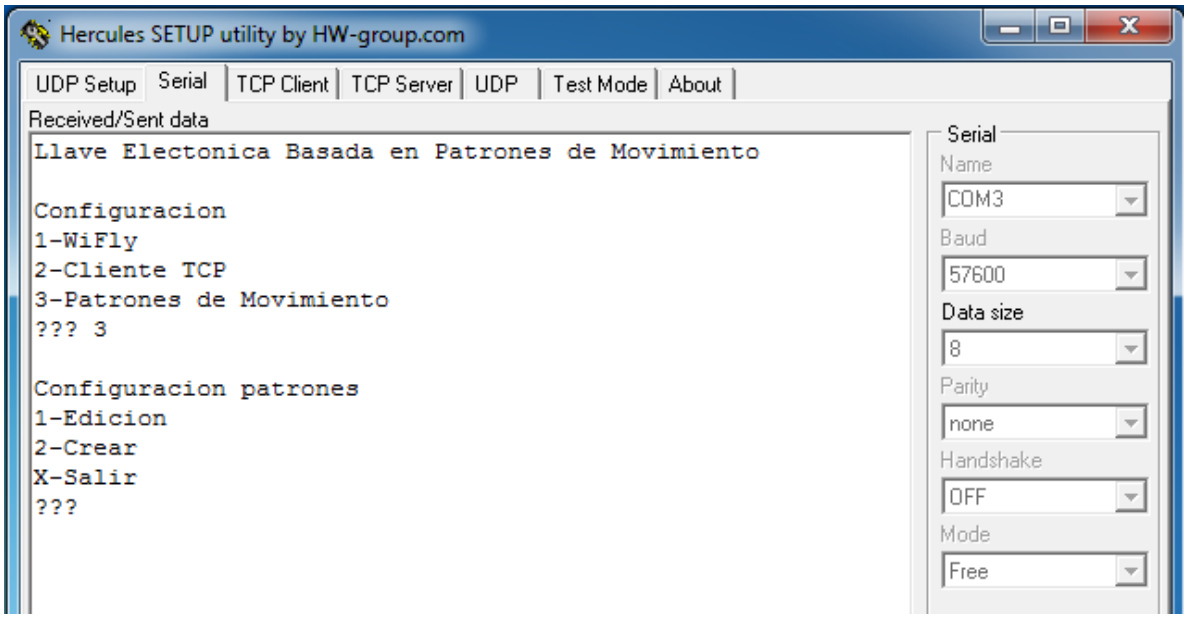



*Figura 31. Menú patrones de movimiento*

La primera selección del submenú permitirá ver el nombre y borrar los patrones almacenados.

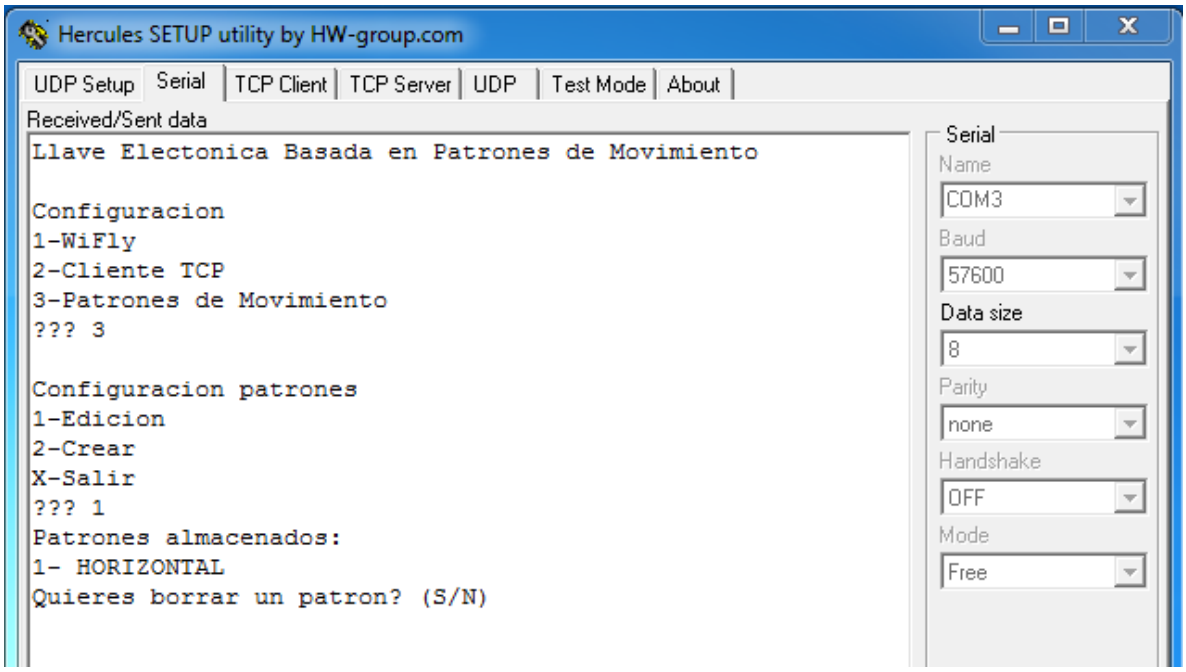



*Figura 32. Edición de los patrones almacenados*

Si se selecciona borrar un patrón nos preguntará por el número del patrón. Este número no es más que una referencia temporal que numera la lista mostrada y no está asociado al patrón en sí.

La segunda selección del submenú de patrones de movimiento permitirá crear nuevos patrones. El sistema admite un máximo de 10, superado este número mostrará un mensaje indicando este hecho.

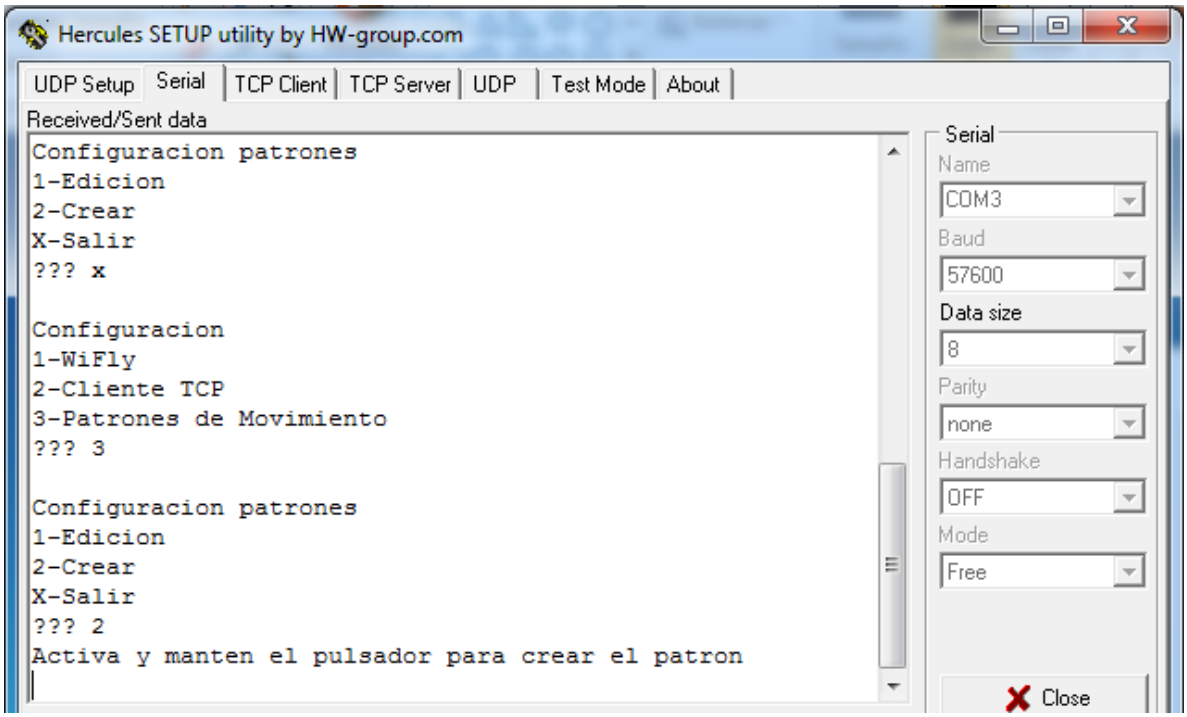



*Figura 33. Creación de un patrón de movimiento*

Al seleccionar esta opción no indica qué tenemos que hacer para crear un patrón. Tenemos que activar el pulsador para empezar y mantenerlo pulsado mientras creamos el patrón. Mientras el sistema esté capturando un patrón el LED *Lectura Patrón* estará parpadeando. Los patrones tienen un límite de movimientos, si nos excedemos de ellos dará por finalizada la creación del patrón. Al finalizar la captura el LED se apagará, y nos pedirá confirmación para guardar el patrón creado y, en caso de aceptar, un nombre para darle al patrón de no más de diez caracteres.

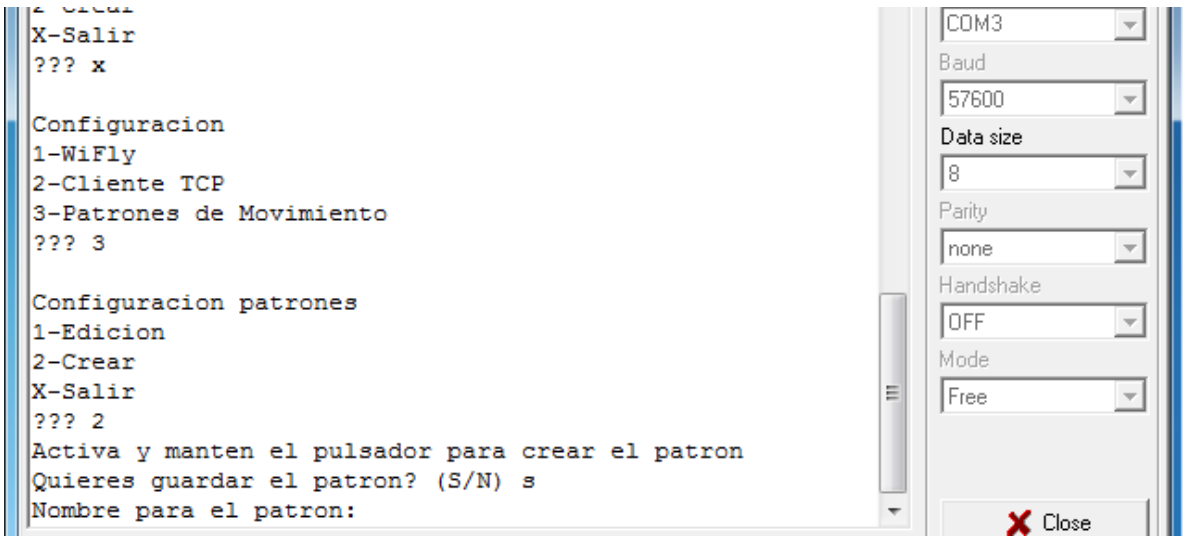



*Figura 34. Asignación de nombre para un patrón*

### 8.1.4 Entrada de una clave de movimiento

Si no estamos configurando el sistema, éste está esperando la entrada de una clave de movimiento. Para introducirla simplemente tenemos que activar el pulsador y, manteniéndolo pulsado, realizar el movimiento. Automáticamente comprobará si el movimiento introducido corresponde con alguno de los patrones que tiene almacenados, en caso afirmativo conectará con el host configurado enviándole en nombre del patrón reconocido.

### 8.1.5 LEDs indicativos

Los LEDs incluye el sistema tienen el siguiente significado:

- **Sistema Activo**

  Fijo: el sistema se ha iniciado correctamente.

- **Conexión WLAN**

  Parpadeando: La conexión a la red no está activa, está intentando conectar. Si permanece por mucho tiempo es indicativo de que los parámetros del WiFly no están correctamente configurados o que no tiene acceso a la WIFI configurada.

  Fijo: La conexión a la red está activa y correcta.

- **Lectura Patrón**

  Parpadeando: Se está introduciendo un movimiento, bien para crear un patrón o bien como entrada de clave de movimiento.

- **Envío a Host**

  Parpadeando: Se ha introduciendo una clave de movimiento, ha sido reconocida y se está enviando la notificación al host. Si permanece por mucho tiempo es indicativo de que no consigue conectar con el host configurado, bien por no estar correctamente configurados los parámetros del cliente TCP o bien porque el host no está accesible.

## 8.2 Ejecución y compilación

Para la compilar y cargar el proyecto en el LPC es necesario tener instalado el LPCXpresso.

Las siguientes indicaciones están realzadas con la versión LPCXpresso v7.0.0 [Build 92] [2014-02-20].

### 8.2.1 Importación del proyecto

Los pasos a seguir para importar el proyecto son los siguientes:

a) Guardar el archivo *FreeRTOS_workspace_UOC.zip* en alguna carpeta sin descomprimir.
b) Seguir los pasos mostrados en las figuras.

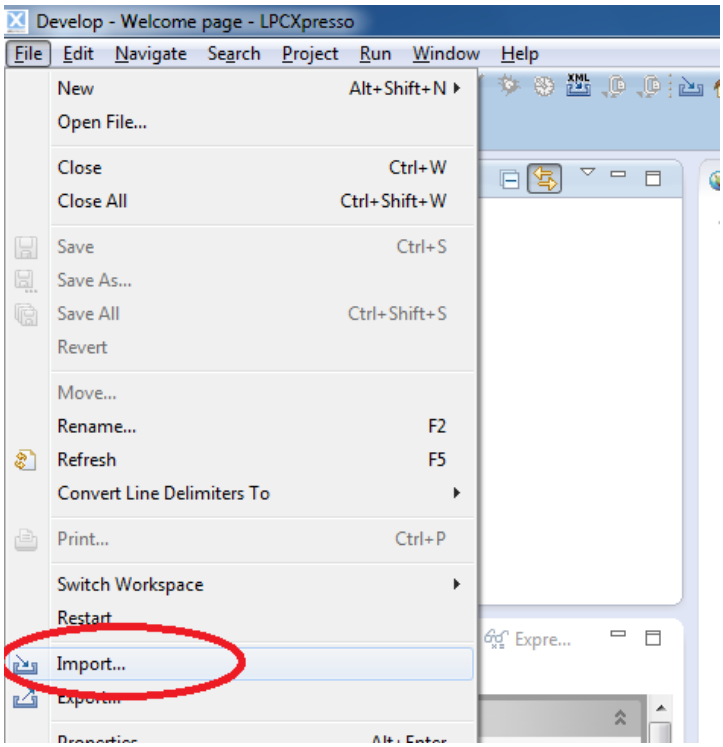



*Figura 35. Importar proyecto LPCXpresso (1)*

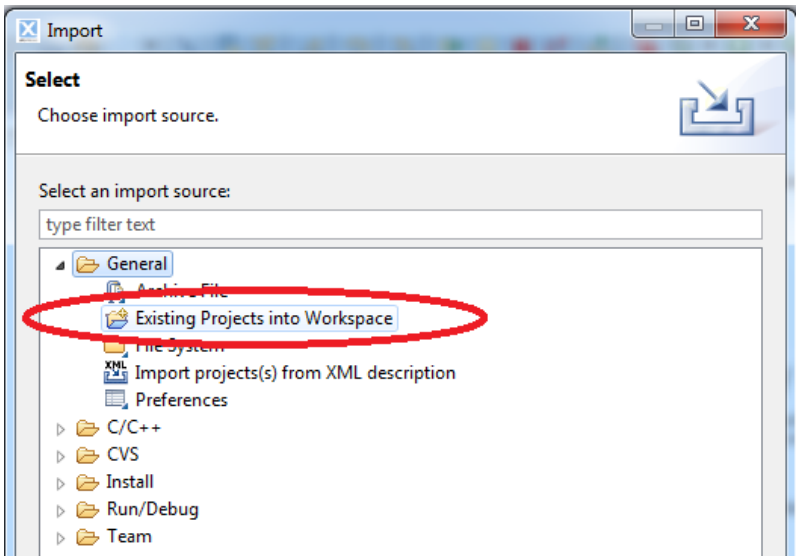



*Figura 36. Importar proyecto LPCXpresso (2)*

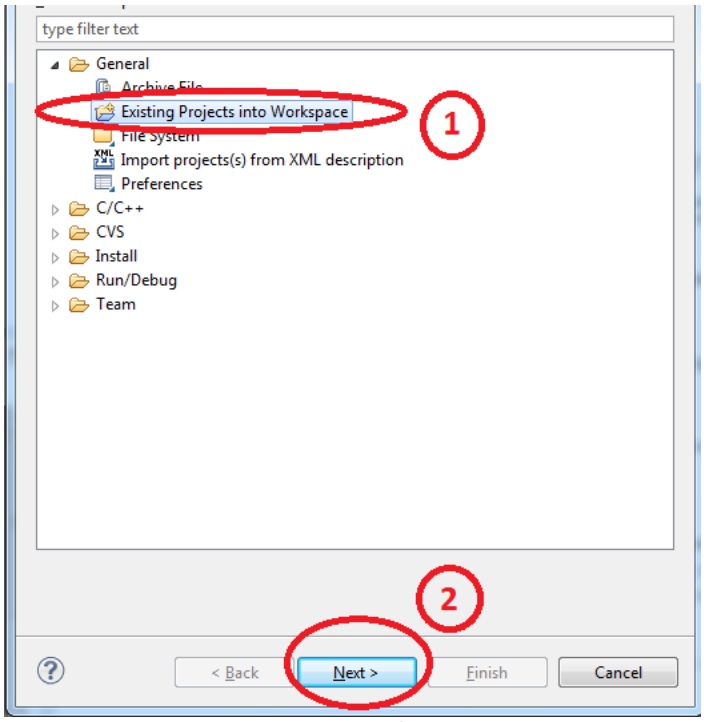



*Figura 37. Importar proyecto LPCXpresso (3)*

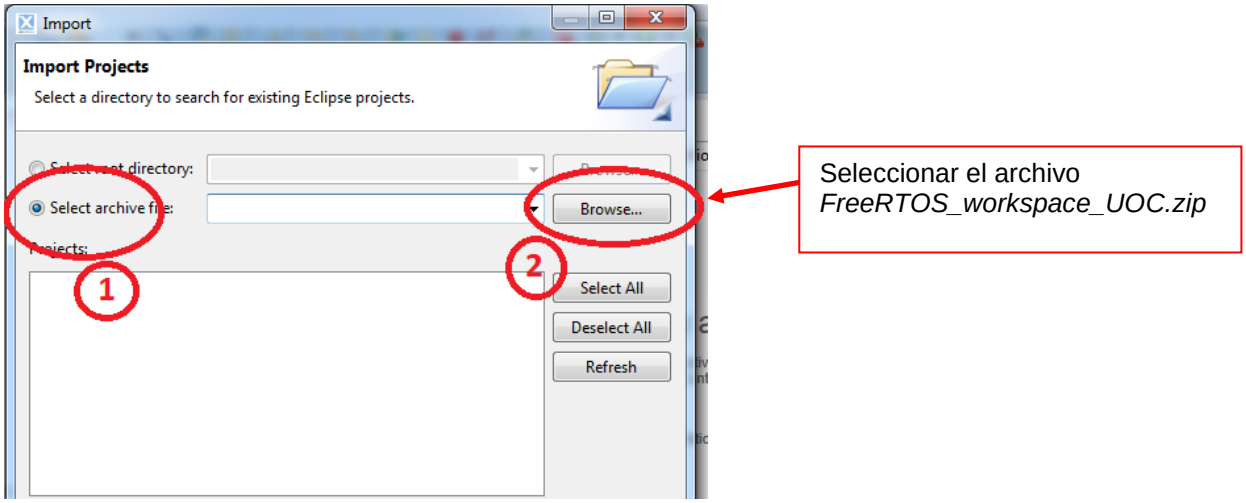



*Figura 38. Importar proyecto LPCXpresso (4)*

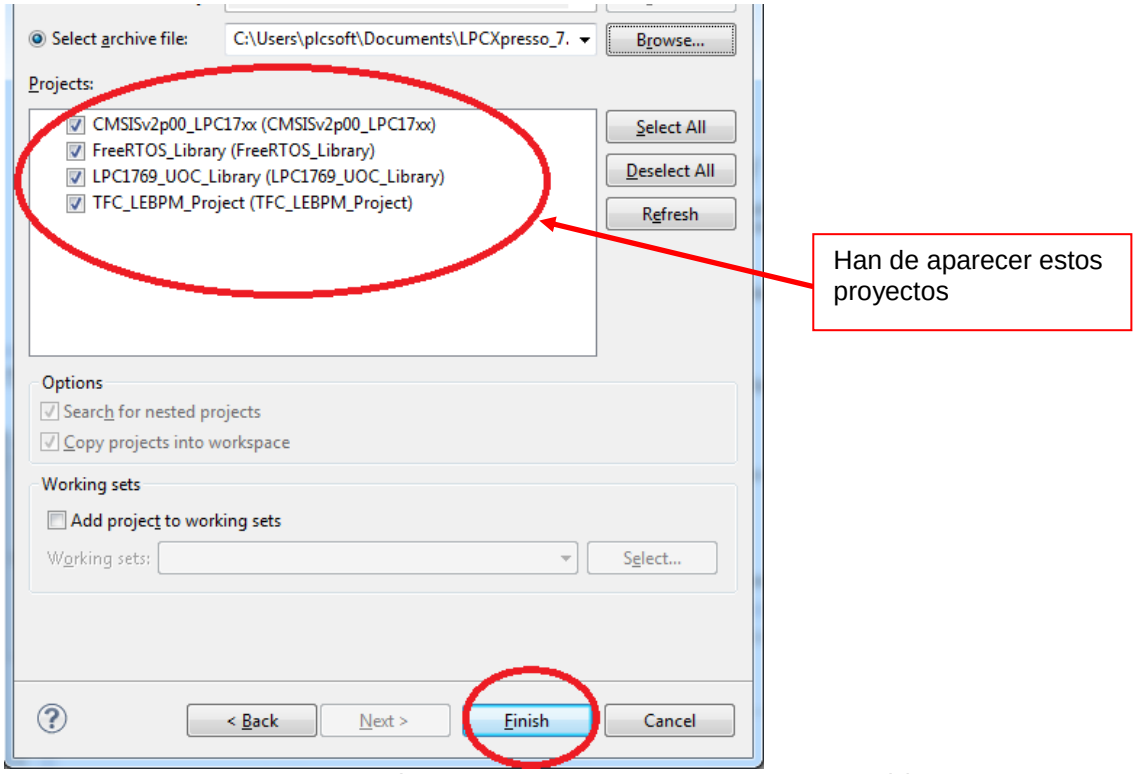



*Figura 39. Importar proyecto LPCXpresso (5)*

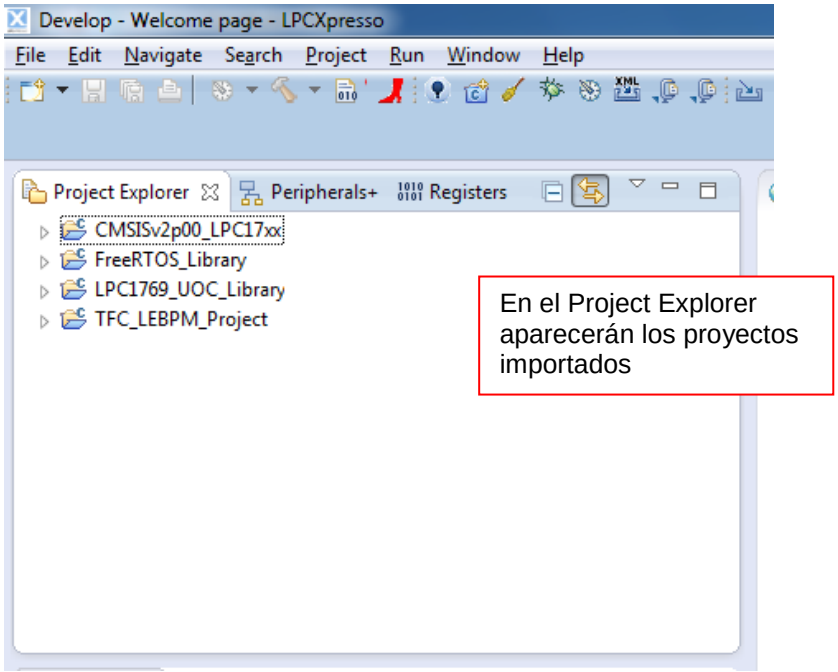



*Figura 40. Importar proyecto LPCXpresso (6)*

### 8.2.2 Compilación del proyecto

Los pasos a seguir para compilar el proyecto son los siguientes:

a) Seguir los pasos mostrados en las figuras.

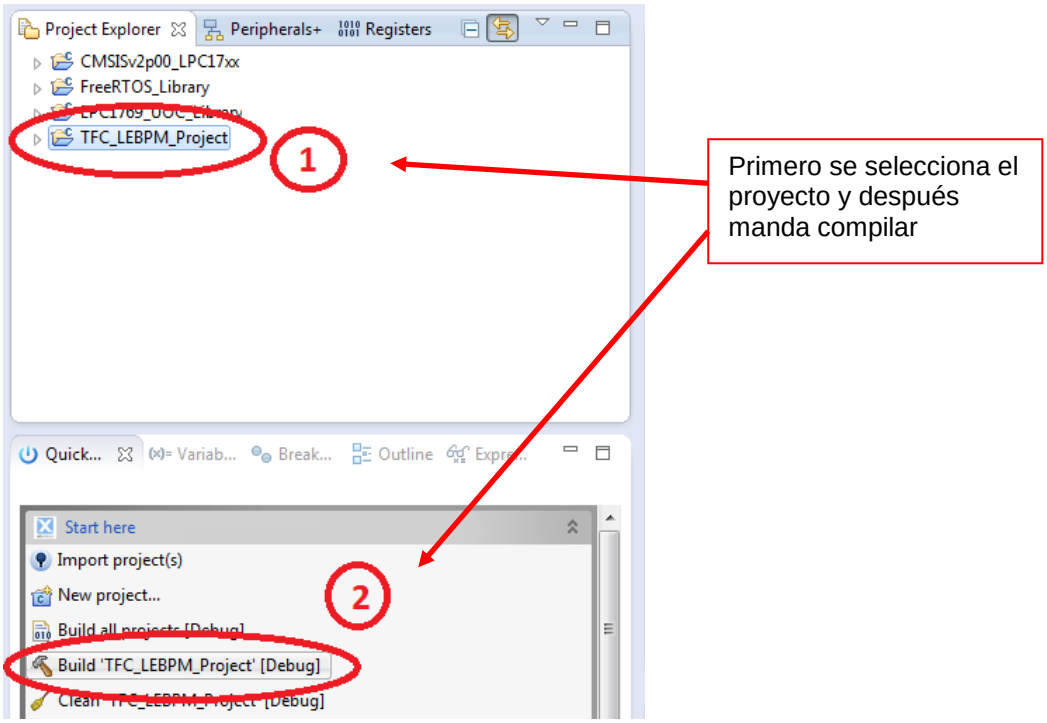



*Figura 41. Compilar proyecto LPCXpresso (1)*

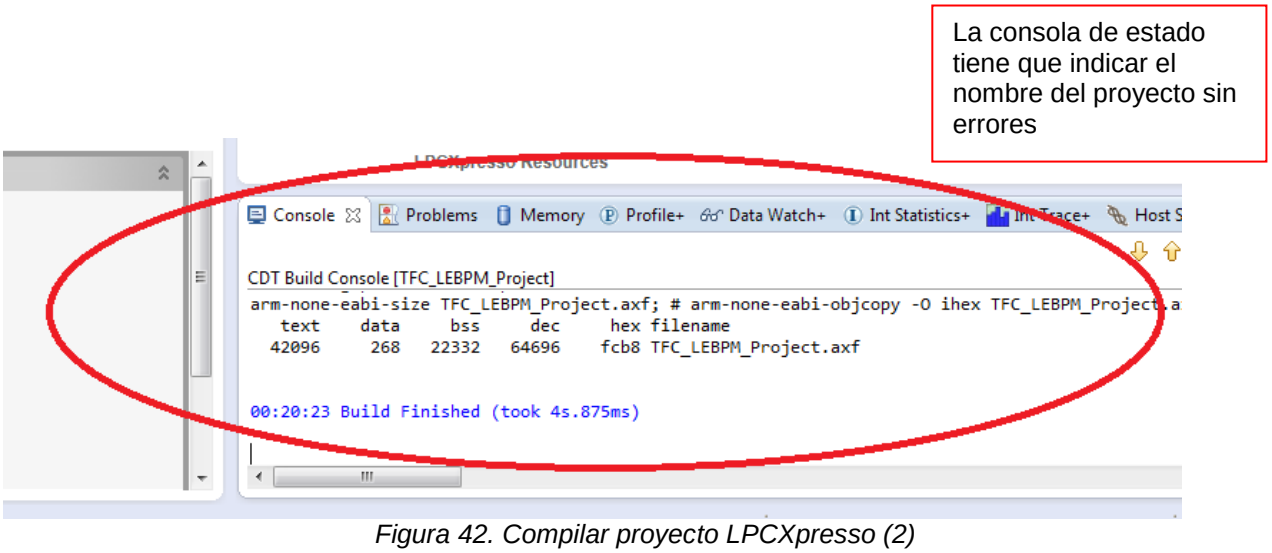


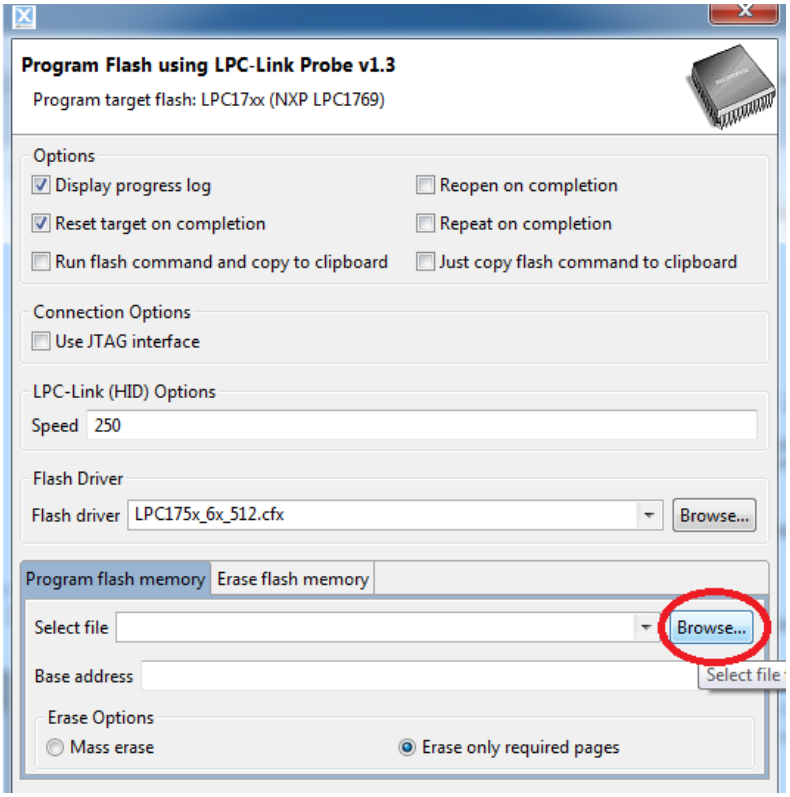

*Figura 42. Compilar proyecto LPCXpresso (2)*

### 8.2.3 Cargar del proyecto

Los pasos a seguir para cargar el proyecto son los siguientes:

a) Conectar el LPC al PC
b) Seguir los pasos mostrados en las figuras.

*Figura 43. Cargar el proyecto en el LPC (1)*




Buscar el archivo *TFC_LEBPM_Project.axf* dentro del directorio *debug* del Workspace

*Figura 44. Cargar el proyecto en el LPC (2)*

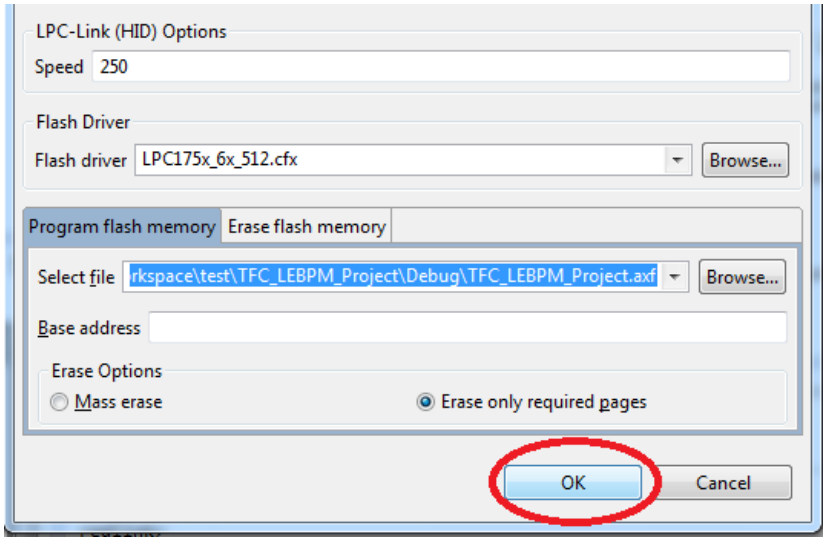



*Figura 45. Cargar el proyecto en el LPC (3)*

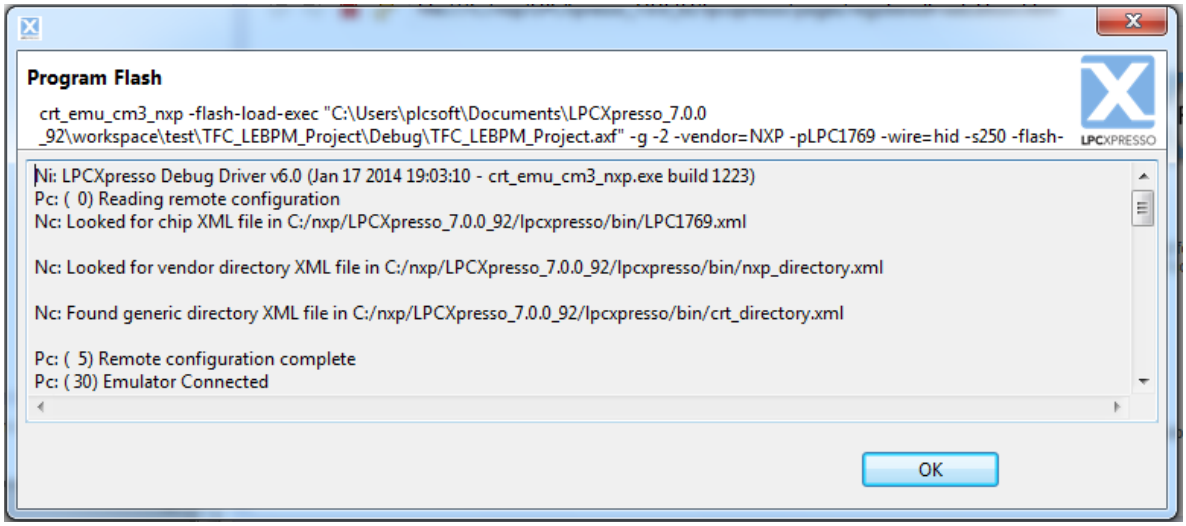



*Figura 46. Indicación de final de cargar del proyecto en el LPC*